Escola Politècnica Superior de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FIN DE CARRERA

TÍTULO DEL TFC: Predicción de tráfico en redes IP

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática

AUTOR: Santiago Escriche Fernández

DIRECTOR: David Rincón Rivera

FECHA: 27 de enero de 2011

Título: Predicción de tráfico en redes IP

Autor: Santiago Escriche Fernández

Director: David Rincón Rivera

Fecha: 27 de enero de 2011

Resumen

A lo largo de los últimos años, las técnicas de pronóstico estadístico se han ido aplicando a la predicción del comportamiento de las redes IP con un doble objetivo. Por un lado, está el objetivo de predecir la carga futura de enlaces individuales o rutas aisladas y así anticipar la aparición de situaciones críticas. Como es de esperar, el hecho de conocer de antemano la aparición de problemas permite solucionarlos antes de que éstos lleguen a producirse. El otro objetivo es establecer una mejor planificación del crecimiento de la infraestructura de red mediante los resultados predictivos que se han obtenido. En este sentido, es más eficiente tratar los datos globalmente, por ejemplo, a partir de las matrices de tráfico.

Este documento presenta una introducción a la utilización de dos de las técnicas de pronóstico estadístico más comunes y más extensamente utilizadas. La primera técnica que se estudiará es el análisis de series temporales basado en los modelos Box-Jenkins o ARIMA. Esta técnica desarrollada en 1970, permite la generación de modelos eficientes adecuados para la predicción de una única serie temporal. La aplicación será el estudio de la carga de los enlaces de una red IP.

La segunda técnica utilizada en éste documento es el análisis de componentes principales (PCA). Este método proporciona modelos genéricos que ofrecen un pronóstico aproximado para una serie particular. No obstante, los modelos generados con este método son útiles para representar el comportamiento de más de una serie con un único modelo y se pueden aplicar en el modelado conjunto de enlaces y en el estudio de matrices de tráfico.

Este trabajo tiene dos objetivos fundamentales. El primero de ellos, es aplicar las técnicas estudiadas al caso de una red real, con el propósito de comprobar la calidad de las predicciones y, en segundo lugar, realizar una valoración de la utilidad de estas técnicas para la planificación del crecimiento de una red.

Title: Traffic prediction in IP networks

Author: Santiago Escriche Fernández

Director: David Rincón Rivera

Date: January, 27th 2011

Overview

During the last years, statistical forecasting techniques have been applied to the prediction of the behavior of IP networks with a double objective. On the one hand, the aim is to predict the future load of individual links and to anticipate critical situations such as overload. Predicting critical situations in advance allows solving them before it is too late. Another aim is to establish a better planning of the growth of the network's infrastructure by the use of the predictive results that have been obtained. In the latter case, it makes sense to analyze the data globally (for example, using traffic matrices).

This document presents an introduction to the use of two of the most common and extensively used techniques of statistical forecast. The first technique that will be studied is the time series analysis based on the Box-Jenkins or ARIMA models. This technology was developed in 1970 and allows the generation of efficient models adapted for the prediction of a unique time series. Its application will be the study of the link load in an IP network.

The second technique used in this document is the principal component analysis (PCA). This method provides generic models that offer an approximate forecast for a particular time series. Nevertheless, the models generated with this method are useful to represent the behavior of a group of time series with a unique model and can be applied to the joint modeling of several link load series, or to traffic matrices.

This work has two fundamental aims. The first of them is to apply the studied techniques to a real and operational network, with the intention of verifying the quality of the predictions and, in second place, to perform an evaluation of the usefulness of these techniques for the planning of a network's growth.

ÍNDICE

INTRODUCCIÓN

Históricamente, las técnicas de pronóstico estadístico se han utilizado sobre todo para realizar predicciones del comportamiento del mercado financiero, como la previsión de tendencias al alza y a la baja de la bolsa. No obstante, en los últimos años, el uso de estas técnicas se ha extendido a la predicción del comportamiento del tráfico IP y se ha convertido en una disciplina especialmente valiosa para optimizar los recursos de una red.

El uso de estas técnicas para realizar pronósticos del comportamiento futuro de una red tiene dos ventajas inmediatas. Por una parte, proporciona la posibilidad de aplicar un uso eficiente a la capacidad de carga de una red, ya que permite anticipar situaciones críticas y solucionarlas por adelantado. Por otra parte, los resultados predictivos se pueden usar para obtener una aproximación del crecimiento futuro de la red y así contribuir a la planificación de su infraestructura.

El objetivo de este trabajo es la aplicación de estas técnicas a datos de tráfico correspondientes a una red operativa: RedIRIS, la red académica que une universidades y centros de investigación españoles con el resto de Internet, con el propósito de comprobar la calidad de la predicción y poder realizar una evaluación de la utilidad de las técnicas estudiadas para la planificación del crecimiento de una red. Este trabajo también tiene el objetivo de proporcionar una base teórica y práctica sobre la que desarrollar futuros proyectos orientados al estudio extensivo de series temporales aplicadas al ámbito de las redes IP.

A lo largo de éste documento se prima la utilización de modelos sencillos, ya que hay varias razones para escoger un modelo con pocos parámetros:

- En general, hay un compromiso entre la fidelidad del modelo y su capacidad predictiva: si tiene muchos parámetros obtendremos un modelo muy fiable para ese conjunto de datos, pero que no nos servirá como modelo predictivo porque será demasiado específico y será difícilmente aplicable a otras situaciones.
- Si el modelo dispone de pocos parámetros será mucho más fácil asignarles sentido físico, de manera que se incrementará el control que tenemos sobre el modelo.

En el primer capítulo se puede encontrar una introducción al análisis de series temporales, una técnica que permite la generación de modelos predictivos para series temporales. En este capítulo se definen también las series temporales y las características que han de satisfacer para poder ser utilizadas en el análisis.

En el segundo capítulo se detalla la descripción de RedIRIS, su topología y la composición de los datos proporcionados por esta red.

En el tercer capítulo se explican las herramientas necesarias para realizar un análisis de series temporales basado en los modelos ARIMA o de Box-Jenkins.

En el cuarto capítulo se lleva a cabo la aplicación del método ARIMA a una serie real con el objetivo de predecir su comportamiento futuro.

En el quinto capítulo se detalla la metodología del análisis de componentes principales y su utilización como medio de obtención de modelos genéricos para la predicción de conjuntos de datos que comparten un mismo patrón.

En el sexto capítulo se aplica el método del análisis de componentes principales a un caso real, con el objetivo de obtener un modelo genérico que permita predecir el comportamiento de más de una serie.

Finalmente, se presentan las conclusiones y las líneas futuras de desarrollo a partir de los resultados obtenidos en este proyecto.

CAPÍTULO 1. INTRODUCCIÓN AL ANÁLISIS DE SERIES TEMPORALES

1.1. Las Series Temporales

1.1.1. Definición de las Series Temporales

Una serie temporal es un conjunto ordenado de observaciones $\{z_1,\ldots, z_t,\ldots, z_n\}$ obtenidas en intervalos regulares de tiempo, donde z_t es la observación de la variable de interés en el instante t. El instante temporal t puede ser un año, trimestre, mes, semana, etc. y determina la frecuencia de observación (anual, trimestral, mensual, semanal, etc.). Así, las observaciones de la serie se obtienen en instantes equidistantes de tiempo y se descartan así las series temporales compuestas.

1.1.2. El concepto de homogeneidad

Para que el análisis de una serie temporal conduzca a conclusiones acertadas no basta con utilizar las técnicas apropiadas, sino que también es imprescindible que los datos sean comparables y no lo serán nunca si no son homogéneos. Si a lo largo de la serie cambia la metodología de observación, se cambian las definiciones, se modifica la población de referencia, etc., el resultado será una serie temporal compuesta por un conjunto de valores no comparables por ser muy heterogéneos.

La homogeneidad se pierde, de todos modos, de una forma natural con el transcurso del tiempo, ya que cuando las series son muy largas no hay garantía de que los datos iniciales y finales sean comparables. La necesidad de que las series no sean muy largas para que sus datos no pierdan su homogeneidad, entra en conflicto con el objetivo más elemental de la Estadística, que es el de detectar regularidades en los fenómenos de masas.

1.1.3. Clasificación de las series temporales

Las series temporales se pueden clasificar en dos grandes grupos según su predictibilidad:

- Las series deterministas, que permiten realizar una predicción exacta del próximo valor de la serie a partir de los valores pasados.
- Las series no deterministas, aleatorias o estocásticas, que no permiten anticipar un resultado futuro con total certeza.

Es sobre estas últimas sobre las que se aplica el análisis de series temporales, que comprende métodos que ayudan a interpretar este tipo de datos. Estos métodos extraen las relaciones subyacentes y las utilizan para extrapolar y

predecir el comportamiento futuro de la serie. La posibilidad de aplicar estos procedimientos a series obtenidas en el mundo físico (mayoritariamente aleatorias), ha hecho que el análisis estadístico de series temporales sea utilizado con profusión en muchas áreas de la ciencia, como en física, ingeniería y economía.

1.2. Los tipos de procesos

1.2.1. Los procesos estocásticos

El análisis de series temporales se basa en el hecho de que, a pesar de que el valor futuro de una serie temporal no sea predecible con total exactitud, éste tampoco es completamente aleatorio y presenta regularidades que hacen posible su predicción dentro de unos umbrales de error. Por lo tanto, si se encuentran patrones de regularidad en distintas secciones de una serie temporal, ésta se puede describir también mediante modelos basados en distribuciones de probabilidad. Así, la secuencia ordenada de variables aleatorias Y_t y su distribución de probabilidad asociada conforman lo que se denomina un proceso estocástico, el modelo matemático para una serie temporal.

1.2.2. Los procesos estacionarios

Un concepto importante que se puede encontrar dentro del ámbito de los procesos estocásticos, es el de los procesos estacionarios. De una manera informal, una serie es estacionaria en sentido amplio o débil cuando se encuentra en equilibrio estadístico. Esto equivale a decir que la media y la varianza se mantienen constantes a lo largo del tiempo y por lo tanto, la serie no presenta tendencias, que son los cambios a largo plazo de la media. Además, tampoco pueden presentar efectos estacionales, es decir, que el comportamiento de la serie sea parecido para instantes de tiempo periódicos.

1.2.3. El ruido blanco

El ruido blanco o puramente aleatorio representa un buen ejemplo de proceso estacionario, ya que es una secuencia de variables aleatorias $\{a_t\}$ mutuamente ortogonales con media cero, varianza constante y covarianzas nulas entre valores correspondientes a distintos valores de t.

El ruido blanco se utiliza en el análisis de series temporales para representar las innovaciones en una serie, ya que modela a la perfección el carácter impredecible de las variaciones aleatorias de una serie.

CAPÍTULO 2. REDIRIS

2.1. Introducción

RedIRIS [1] es la red académica y de investigación española y proporciona servicios avanzados de comunicaciones a la comunidad científica y universitaria nacional (actualmente cuenta con más de 350 instituciones afiliadas, entre las que se encuentra la UPC). Está financiada por el Ministerio de Ciencia e Innovación, e incluida en su mapa de Instalaciones Científico Tecnológicas Singulares. Se hace cargo de su gestión la entidad pública empresarial Red.es, del Ministerio de Industria, Turismo y Comercio.

2.2. Topología

RedIRIS tiene una red troncal de telecomunicaciones denominada RedIRIS-10 que comunica las instituciones universitarias y de investigación españolas entre sí, a través de las redes académicas autonómicas:

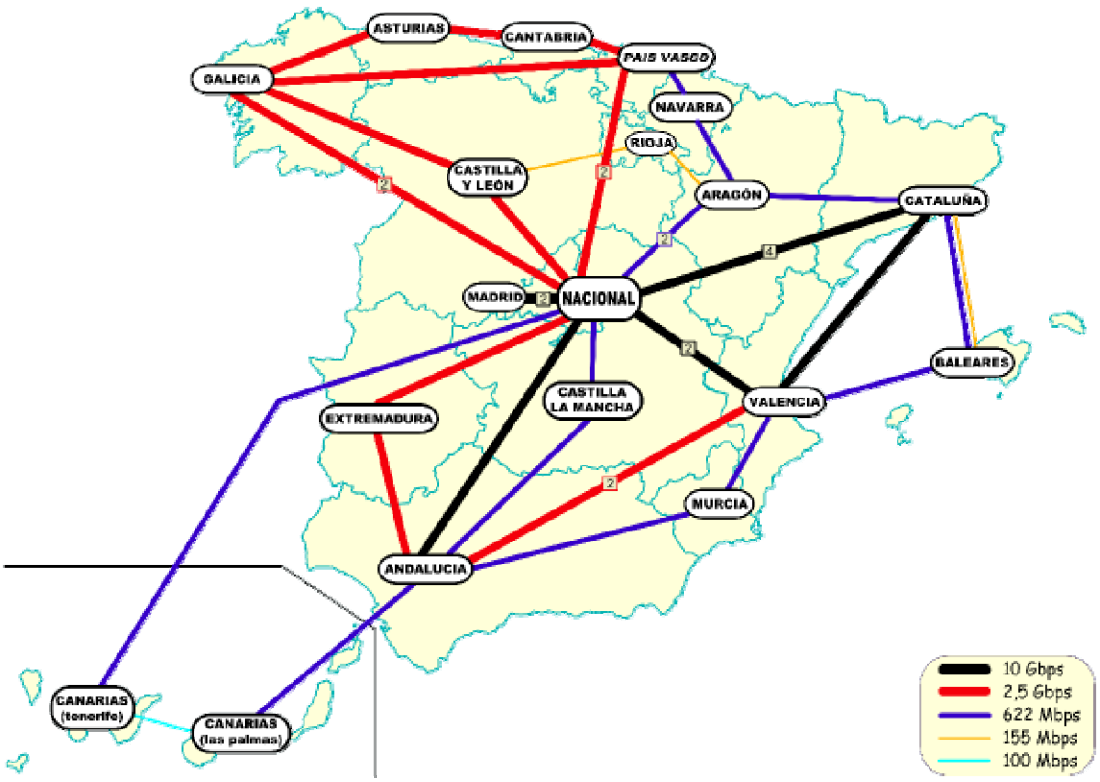


Fig. 2.1 Topología de RedIRIS-10

RedIRIS-10 proporciona acceso a la red de investigación mundial a través de la red pan-europea GÉANT [2], una infraestructura de red de fibra oscura con un

punto de presencia por país, que interconecta a 33 redes nacionales de investigación. GÉANT es una red híbrida donde se soportan servicios de conmutación de circuitos y conmutación de paquetes. Proporciona además el acceso a las redes de investigación de otras zonas del mundo como Internet2 (USA), Canarie (Canadá), RedCLARA (América Latina), EUMEDCONNECT (Norte de África), UbuntuNet (Este y Sur de África), TENET (Sur de África), TEIN2(Asía Pacífica), SINET(Japón), CERNET, CSTNET (en China) o ERNET (India).

La conectividad externa de RedIRIS se complementa con el acceso a la Internet Comercial Global a través de dos salidas a dos proveedores de ámbito internacional. Además, RedIRIS tiene presencia en el punto de intercambio (o IXP, *Internet Xchange Point)* nacional ESPANIX ubicado en Madrid [3] y en el ubicado en Barcelona, CATNIX [4].

2.3. Los datos proporcionados por RedIRIS

Los datos que fundamentan los análisis de todo el trabajo han sido extraídos de las bases de datos donadas por RedIRIS y todos los análisis versan sobre la predicción del tráfico de los enlaces que componen a la red académica y de investigación española.

Las medidas de tráfico que componen las bases de datos se han extraído de la herramienta MRTG, una herramienta gráfica que se utiliza para supervisar el tráfico de interfaces de red. MRTG utiliza el protocolo SNMP (Simple Network Management Protocol) para obtener la información en crudo de la cantidad de bytes que han pasado por cada router que pertenece a la red, distinguiendo entre entrada y salida. Esta información es almacenada en una base de datos gestionada por RRDtool (*Round Robin Database tool)* [5,6], una aplicación que trabaja con una base de datos que sigue un modelo Round Robin, es decir, que selecciona todos los elementos de la base de datos comenzando por el primer elemento de la lista hasta llegar al último y vuelve a empezar de nuevo desde el primer elemento [7]. A partir de esta información y de forma separada, se generan los informes y gráficas.

La recolección de información y por lo tanto, la creación de un nuevo registro en la base de datos, está fijada a cinco minutos por defecto. No obstante, internamente se aglutinan resultados para disponer de un historial más extenso, a costa de la ampliación del lapso temporal asociado a cada registro y la consecuente pérdida de granularidad. Así, se suelen definir registros de 300 segundos (5 minutos), 1800 segundos (media hora), 7200 segundos (dos horas) y de 86400 segundos (un día), en función del lapso temporal que recogen. Además, se realizan almacenamientos de los registros máximos, mínimos, medias, etc. para cada una de estas clases de registro.

Hay que destacar que la configuración de las bases de datos RRD limita la extensión del número de registros a un número fijo y todos los valores añadidos con posterioridad reescriben los datos más antiguos (buffer circular). Esto significa que para un mismo número de registros, aquellos registros que

contemplan un lapso superior de tiempo también proporcionan una visión más lejana del historial de la serie. Así, para las bases de datos de RedIRIS:

- Los registros de 300 s recogen los datos de las dos últimas horas
- Los registros de 1800 s recogen los datos de los quince últimos días
- Los registros de 7200 s recogen los datos de los dos últimos meses
- Los registros de 86400 s recogen los datos de los dos últimos años

MRTG [8] utiliza la información de los registros para generar un informe en formato HTML con gráficas que proveen una representación visual de la evolución del tráfico a lo largo del tiempo. Esta aplicación está implementada en la página web de RedIRIS bajo el nombre de *Weathermap* [9] y permite observar el flujo de datos de cada enlace en tiempo real:

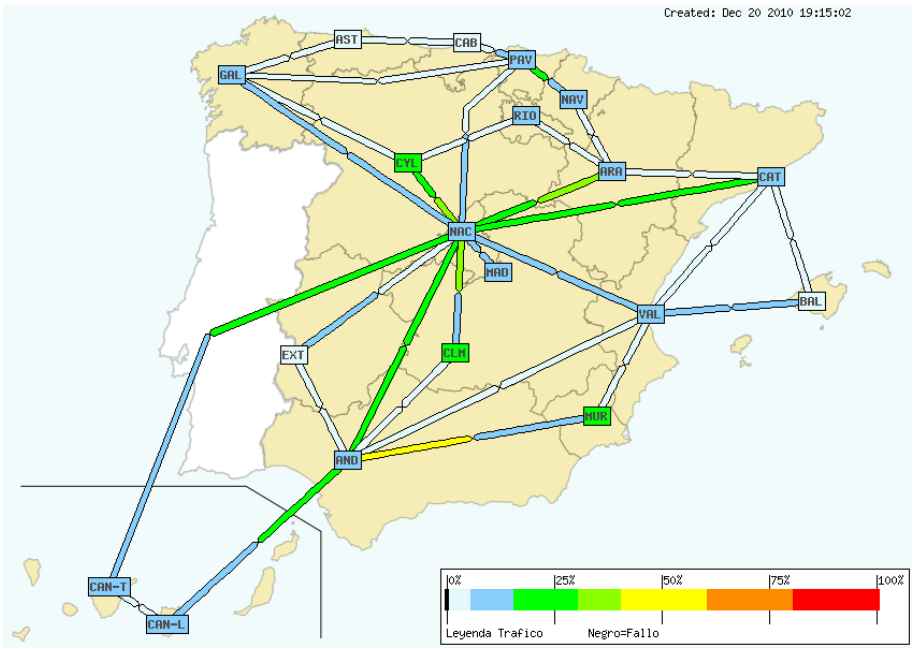


Fig. 2.2 Mapa global de RedIRIS proporcionado por Weathermap

Como se puede observar en la figura anterior, el mapa global de RedIRIS muestra la utilización asociada a cada enlace mediante el código de colores indicado en la leyenda. Cada enlace es bidireccional y de ahí la posible aparición de distintos porcentajes de utilización en uno y otro sentido. Si hacemos clic en uno de estos enlaces, obtendremos la representación gráfica del flujo de datos diario, semanal, mensual y anual asociados al enlace seleccionado. En la Fig. 2.3, que muestra el comportamiento del enlace Barcelona-Madrid (en las dos direcciones, con colores verde sólido y azul) se pueden apreciar las diferentes periodicidades (día, semana, año) de la carga del enlace.

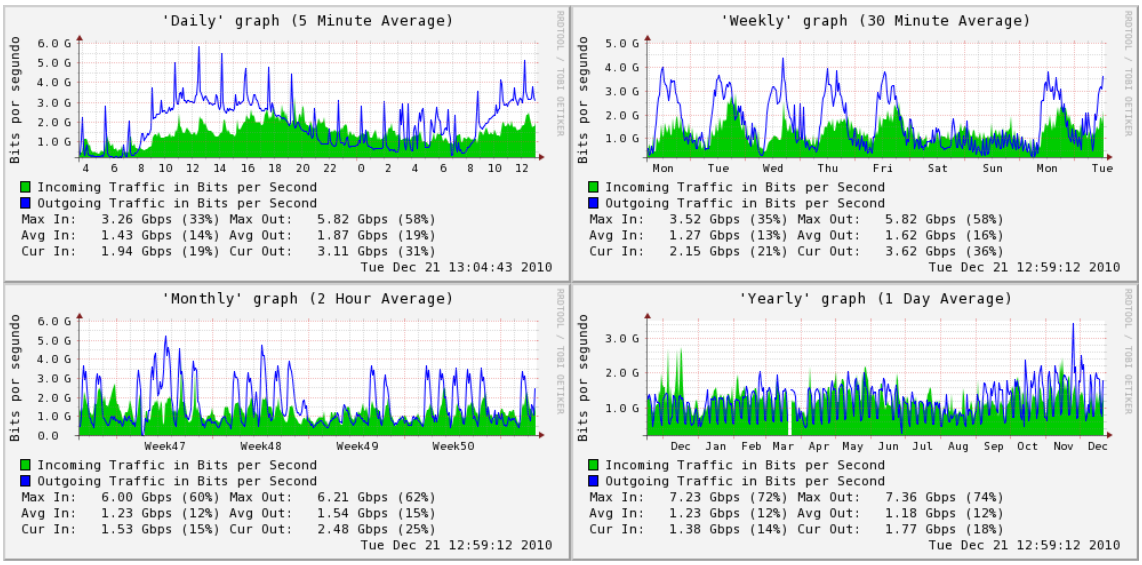


Fig. 2.3 Gráfico de los datos del enlace nac-cat de RedIRIS-10

2.4. Las Series Temporales

Como hemos indicado anteriormente, los registros recogen el historial de las últimas 2 horas, 15 días, 2 meses y 2 años. No obstante, para el análisis sólo se han utilizado los registros referentes a los últimos 15 días, ya que son los registros que permiten el mejor compromiso entre alcance predictivo y granularidad para predicciones a corto y medio plazo.

En concreto, los registros utilizados recogen las medias de flujo de los enlaces entre las 23:30:00 horas del 29/03/2010 hasta las 13:00:00 horas del 13/04/2010, ambos incluidos.

Si se consulta un calendario de los meses de marzo y abril de 2010, se puede observar que sólo los días del 05/04/2010 al 09/04/2010, el 12/04/2010 y el 13/04/2010 son laborables, mientras que el resto corresponden a fines de semana y días festivos de semana santa. Es sobre los registros referentes a estos días sobre los que resulta interesante aplicar el análisis de series temporales, ya que los días laborables son los que presentan una mayor utilización a la red y unas medidas más ajustadas de la carga total que suele presentar cada enlace. Así pues, las series se han construido mediante la sucesión de días laborables, descartando los festivos intermedios que distorsionaban el comportamiento regular de la red.

Estos datos son una primera aportación de RedIRIS a los esfuerzos de la aplicación de las técnicas de pronóstico estadístico al ámbito de las redes IP y constituyen una toma de contacto con la realidad de la red académica y de investigación española. Por esta razón, los registros utilizables son muy reducidos y dejan poco margen para establecer y verificar predicciones de largo alcance. No obstante, estas pruebas iniciales sientan una base sólida sobre la que aplicar futuros análisis sistemáticos en otros trabajos o TFCs.

2.5. Tipos de series

La topología de RedIRIS y su política de encaminamiento condicionan irremediablemente el patrón de comportamiento de los enlaces que la constituyen. Así, se pueden distinguir dos tipos principales de enlace:

- Los enlaces “primarios”, definidos así porque el tráfico que transcurre por ellos es estable y presenta una tendencia clara.

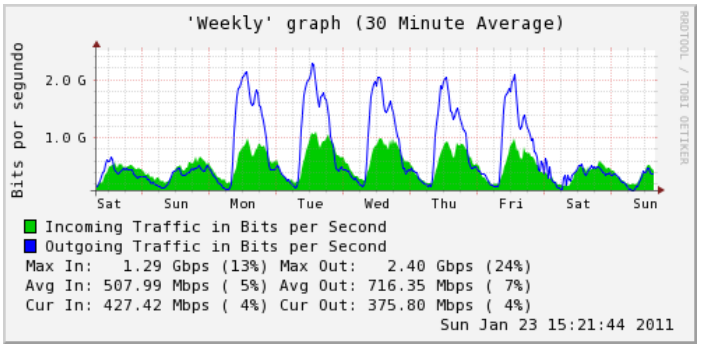


Fig. 2.4 Gráfico semanal del enlace nac-val (Madrid Valencia) de RedIRIS-10

- Los enlaces “de back-up”, en los que el tráfico es más bien a ráfagas, con una tendencia errática, surcada de picos y sin tendencias suaves.

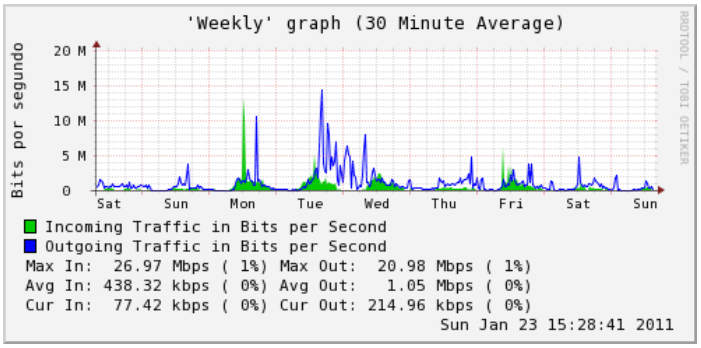


Fig. 2.5 Gráfico semanal del enlace cat-bal1 (BCN-Mallorca) de RedIRIS-10

La regularidad en los datos es una característica muy apreciada en el análisis estadístico, ya que se traduce en predictibilidad. Así, las series de datos de los enlaces primarios serán fácilmente analizables, mientras que las de los enlaces de back-up no lo serán.

Esta situación viene provocada por el carácter radial de la política de encaminamiento de RedIRIS; es decir, los enlaces directos de cada nodo hacia/desde el nodo Nacional son los escogidos como ruta prioritaria, y por tanto llevan una carga elevada y estable, mientras que otros enlaces entran en funcionamiento solo para tráficos directos entre nodos periféricos o cuando fallan los directos. Esto es especialmente visible si se comparan los datos de la topología de RedIRIS (fig. 2.1) con las capturas del porcentaje de utilización

que ofrece el Weathermap (fig. 2.2): los enlaces que provienen del nodo Nacional son los que presentan un tráfico más numeroso. Esta situación se podría mejorar introduciendo algoritmos de balanceo de carga que harían que los enlaces tuviesen un comportamiento más similar y estable, aunque a día de hoy no se utilizan en RedIRIS.

CAPÍTULO 3. EL MÉTODO DE ANÁLISIS ARIMA O DE BOX-JENKINS

3.1. Introducción

Uno de los métodos de análisis de series temporales más ampliamente utilizado es el basado en los modelos Box-Jenkins [10,11], desarrollado a principios de los años 70 del pasado siglo por George E.P. Box, profesor de Estadística de la Universidad de Wisconsin y por Gwilym M. Jenkins, profesor de Ingeniería de Sistemas de la Universidad de Lancaster. Estos modelos también se conocen como modelos ARIMA, las siglas de Modelos Autorregresivos Integrados de Medias Móviles. Una de las ventajas de estos modelos es su gran simplicidad, ya que están constituidos por sumas de términos, frente a los modelos propuestos en la formulación clásica.

En esencia, el método propuesto por Box y Jenkins consiste en proponer un modelo ARIMA válido que describa de la manera más fiel posible a la realidad, la evolución de una variable Y_t de un proceso estocástico en función del pasado de esa variable o de impactos aleatorios que esa variable sufrió en el pasado.

La utilización de modelos ARIMA como estrategia de predicción de series temporales sólo tiene sentido si las características observadas en la serie estudiada, o más correctamente, en el proceso estocástico subyacente, perduran en el tiempo. Esto implica que la serie sobre la que se aplica el análisis ha de cumplir necesariamente la condición de estacionariedad aunque sólo sea de forma débil.

3.2. Los modelos Box-Jenkins básicos

Para modelar los procesos estocásticos subyacentes, se utilizan dos tipos de formas funcionales lineales sencillas: los modelos AR (Modelos Autorregresivos), y los modelos MA (de Medias Móviles).

3.2.1. Los modelos autorregresivos o AR(p)

Un modelo autorregresivo es aquel en el que el valor de la variable para un período t es calculado mediante las observaciones de la variable correspondientes a períodos anteriores (parte sistemática) más un término de error o innovación modelado mediante ruido blanco.

Los modelos autorregresivos se abrevian con las siglas AR tras la que se indica el orden del modelo: AR(1), AR(2),....etc. El orden del modelo expresa el número de observaciones retrasadas de la serie temporal analizada que

intervienen en la ecuación. La expresión general para un modelo autorregresivo de orden p o AR(p) es:

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + a_t \quad \textbf{(3.1)}$$

El valor de la serie Y_t es una combinación lineal de las p observaciones más recientes de sí misma, donde los coeficientes ϕ son los parámetros del modelo asociados a cada observación pasada de Y_t y donde a_t es el valor de la innovación del ruido blanco.

3.2.2. Los modelos de media móvil o MA(q)

Un modelo de media móvil es aquel que explica el valor de la variable para un período t en función de un término independiente y una sucesión de términos de error o innovaciones, correspondientes a períodos precedentes y convenientemente ponderados. Estos modelos se denotan con las siglas MA seguidos del orden entre paréntesis, como en el caso de los modelos autorregresivos.

La expresión general para un modelo de medias móviles con q términos de error MA(q) es:

$$Y_t = \mu + a_t + \theta_1 a_{t-1} + \theta_2 a_{t-2} + \cdots + \theta_q a_{t-q} \quad \textbf{(3.2)}$$

El valor de la serie Y_t es una combinación lineal de las q perturbaciones más recientes, donde los coeficientes θ son los parámetros del modelo asociados a cada innovación pasada de a_t y donde μ es la media de la serie.

3.2.3. La relación entre los modelos AR y MA

Aunque ambos tipos de modelo son necesarios para la generación de modelos predictivos satisfactorios, estos están estrechamente relacionados y de hecho, un modelo de medias móviles puede expresarse como un modelo autorregresivo, ya que si sobre MA(1) se aísla a_t:

$$a_t = Y_t - \theta a_{t-1} \rightarrow a_{t-1} = Y_{t-1} - \theta a_{t-2} \rightarrow a_t = Y_t - \theta Y_{t-1} - \theta^2 a_{t-2} \rightarrow \cdots \quad \textbf{(3.3)}$$
$$\rightarrow a_t = Y_t - \theta Y_{t-1} - \theta^2 Y_{t-2} - \cdots - \theta^j Y_{t-j} \rightarrow$$
$$Y_t = \theta Y_{t-1} + \theta^2 Y_{t-2} + \cdots + \theta^j Y_{t-j} + a_t$$

De manera análoga, un modelo autorregresivo puede reescribirse como un modelo de medias móviles mediante sucesivas substituciones, ya que partiendo de AR(1):

$$Y_t = \phi Y_{t-1} + a_t \rightarrow Y_{t-1} = \phi Y_{t-2} + a_{t-1} \rightarrow Y_t = a_t + \phi a_{t-1} + \phi^2 Y_{t-2} \rightarrow \cdots \quad \textbf{(3.4)}$$
$$\rightarrow Y_t = a_t + \phi a_{t-1} + \phi^2 a_{t-2} + \phi^3 a_{t-3} + \cdots + \phi^j a_{t-j}$$

3.3. Modelos Box-Jenkins avanzados

La realidad demuestra que con sólo los dos modelos básicos, muchos procesos no pueden ser representados. En general:

- Los modelos ARMA(p,q) surgen para resolver el problema de que un modelo simple AR(p) o MA(q) no sea capaz de captar el proceso por completo.
- Los modelos ARIMA(p,d,q) surgen para resolver el problema de que la serie estudiada no sea estacionaria debido a la existencia de una tendencia.
- Los modelos ARIMA(p,d,q)x$(P,D,Q)_s$ surgen para resolver el problema de que la serie estudiada no sea estacionaria debido a la existencia de una estacionalidad y/o de una tendencia.

3.3.1. Modelos Autorregresivos de Media Móvil o ARMA(p,q)

Un modelo autorregresivo de medias móviles es aquel que combina los dos modelos básicos AR(p) y MA(q) para modelar series parcialmente autorregresivas y parcialmente de medias móviles. Estos modelos combinados se abrevian mediante las siglas ARMA(p,q), donde p es el orden de la parte autorregresiva y q es el orden de la parte de media móvil.

La expresión general para un modelo autorregresivo de media móvil ARMA(p,q) es:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + a_t + \theta_1 a_{t-1} + \\ + \theta_2 a_{t-2} + \cdots + \theta_q a_{t-q} + \mu \qquad \textbf{(3.5)}$$

Como el modelo es una combinación de los dos modelos básicos, el valor de la serie Y_t es una combinación lineal de las q perturbaciones más recientes y de las p observaciones más recientes de sí misma.

3.3.2. Modelos Autorregresivos Integrados de Media Móvil o ARIMA(p,d,q)

Una serie temporal $\{Y_t\}$ sigue un modelo autorregresivo integrado de medias móviles si la d-ésima diferenciación (donde d es un número entero) $W_t = \nabla^d Y_t$ es un proceso ARMA estacionario. Si $\{W_t\}$ sigue un modelo ARMA(p,q), se puede decir que $\{Y_t\}$ es un proceso ARIMA(p,d,q). Afortunadamente, para propósitos prácticos, se suele usar una d de 1 o 2.

La ecuación de la forma diferencial de un modelo autorregresivo integrado de media móvil ARMA(p,d,q) es:

$$Y_t = (1+\phi_1)Y_{t-1} + (\phi_2 - \phi_1)Y_{t-2} + \cdots + (\phi_p - \phi_{p-1})Y_{t-p} - \phi_p Y_{t-p-1} + a_t \\ + \theta_1 a_{t-1} + \theta_2 a_{t-2} + \cdots + \theta_q a_{t-q} + \mu \qquad \textbf{(3.6)}$$

3.3.3. Modelos Estacionales Multiplicativos ARIMA o ARIMA(p,d,q)x(P,D,Q)$_s$

Una serie temporal $\{Y_t\}$ es un modelo estacional multiplicativo ARIMA con ordenes no estacionales (regulares) p, d y q, ordenes estacionales P, D y Q y un período estacional s si la serie diferenciada $W_t = \nabla^d \nabla_s^D Y_t$ satisface un modelo ARMA(p,q)x(P,Q)$_s$ con un periodo estacional de s. Así, $\{Y_t\}$ es un proceso ARIMA(p,d,q)x(P,D,Q)$_s$ con periodo estacional s.

En su forma más general el modelo ARIMA(p,d,q)x(P,D,Q)$_s$ puede escribirse como:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_{Ps+p+Ds+d} Y_{t-Ps-p-Ds-d} + a_t + \theta_1 a_{t-1} + \theta_2 a_{t-2} + \cdots + \theta_{Qs+q} a_{t-Qs-q} + \mu \quad \textbf{(3.7)}$$

Esta expresión define al que es el modelo más genérico de los que conforman la familia ARIMA, ya que de hecho, todos los demás son casos particulares de él.

3.4. Condiciones generales de los modelos Box-Jenkins

Cualquier proceso estocástico estacionario puede ser expresado mediante un modelo AR(p) y en consecuencia, también por un modelo MA(q) o una combinación de ambos; siempre y cuando cumpla las siguientes dos condiciones adicionales:

- Hipótesis de Recursividad Temporal: el proceso no debe ser anticipante, o lo que es lo mismo, los valores de una variable en un momento t no pueden depender de los que esta misma toma en t+j, para j>0.
- El proceso ha de ser invertible, lo que supone que la correlación entre una variable y su pasado va reduciéndose a medida que éste se aleja más en el tiempo del momento para el que se considera dicha correlación (conocido también como proceso ergódico). La explicación intuitiva de esta situación deriva del hecho de que, si se especifica una variable en función de ciertos coeficientes que determinen su correlación con los valores pasados de ella misma, los valores de dichos coeficientes deben ser necesariamente inferiores a uno, ya que en caso contrario significaría que el pasado lejano influye más en el presente que el pasado cercano y un proceso de este tipo da origen a series de tiempo explosivas.

3.4.1. Función de autocorrelación

Esta función mide la correlación entre los valores de la serie distanciados un lapso de tiempo j y se utiliza para analizar la estacionariedad de una serie. El coeficiente de correlación para valores distanciados un lapso de tiempo j es:

$$\rho_j = \frac{cov(y_t, y_{t-j})}{\sqrt{var(y_t)}\sqrt{var(y_{t-j})}} \quad \textbf{(3.8)}$$

En la práctica, se suelen calcular y representar juntos un conjunto de lapsos de tiempo consecutivos, con el objetivo de observar su evolución. Se suele utilizar para determinar el orden de los modelos MA(q), la importancia de los residuales o detectar patrones repetitivos dentro de una serie, como por ejemplo, la periodicidad de una estacionalidad.

Para ayudar a determinar qué correlaciones son significativas, se suelen establecer unos umbrales en ±2/√n, donde n es la longitud del registro (número de puntos) de la serie analizada. Estos umbrales establecen los límites superior e inferior de una zona crítica que contiene aquellas correlaciones prácticamente nulas (con un nivel de significancia del 5%). Esta aproximación parte de la asunción de que la longitud del registro es moderadamente larga (n mayor a 30) y que el proceso subyacente es una distribución normal multivariada.

3.4.2. Función de autocorrelación parcial

Para calcular el coeficiente de autocorrelación parcial de orden k, se calcula la correlación entre parejas de valores separados esa distancia pero eliminando el efecto debido a la correlación producida por retardos anteriores a k. Así, la función de autocorrelación parcial proporciona la relación directa que existe entre observaciones separadas por k retardos.

Igual que sucedía con la función de autocorrelación, en la práctica se suelen calcular y representar juntos un conjunto de lapsos de tiempo consecutivos, con el objetivo de observar su evolución. Esencialmente, este gráfico se utiliza para determinar el orden de los modelos AR(p).

3.5. El procedimiento de análisis

El método de análisis de series temporales propuesto por Box y Jenkins [11] está constituido por los siguientes tres pasos: especificación de modelos, estimación de parámetros y diagnóstico del modelo. Una vez superados estos pasos, se suele llevar a cabo el paso adicional que da sentido al análisis: la predicción.

3.5.1. Especificación de modelos

Como se ha indicado anteriormente, la aproximación a los procesos estocásticos con modelos Box-Jenkins está restringida a aquellos procesos que cumplan, al menos de forma débil, la condición de estacionariedad. Así pues, el primer paso del análisis consistirá en comprobar si la serie de observaciones es una serie estacionaria y en caso de no serlo, convertirla en una que sí que lo sea. El método más sencillo e intuitivo para detectar tendencias o

estacionalidades que pongan de manifiesto que la serie es no-estacionaria, es la inspección de su representación gráfica.

El procedimiento para eliminar una tendencia, consiste en aplicar sucesivas diferenciaciones a la serie hasta que ésta desaparezca de la serie resultante y la media de la serie no varíe a largo plazo. Cada sucesiva diferenciación se representa incrementando en uno el grado “d” del modelo.

En algunas series puede aparecer una tendencia logarítmica superpuesta a la tendencia lineal. El signo más destacable de esta situación es la observación de un incremento de la variabilidad para los valores más elevados de la serie. Para estos casos, es recomendable aplicar una transformación logarítmica a la serie que la elimine como paso previo a la eliminación de la tendencia lineal sobre la que se superpone.

En caso de que el sistema presente una estacionalidad, hay que encontrar su período, es decir, cada cuántos instantes de tiempo el comportamiento de la serie se repite. En estos casos resulta especialmente valioso conocer la frecuencia de observación, ya que puede ayudar a precisar con más exactitud el período de la estacionalidad. Para comprobar la existencia de una estacionalidad y verificar cuál es su período, se puede aplicar la función de autocorrelación sobre la serie. Si la serie presenta picos destacables en los retrasos correspondientes al período y a sus múltiplos, la estacionalidad será real y tan significativa como decisivos sean los picos. Una vez encontrado el período, hay que aplicar una diferenciación sobre el retraso correspondiente. Si se aplica esta transformación, el modelo será un SARIMA donde el grado “D” se ha de establecer a uno y se ha de indicar el período de la estacionalidad en el grado “s”.

Las tendencias y estacionalidades hay que atacarlas por orden de claridad y en función de nuestro conocimiento sobre la naturaleza de los datos que se están tratando. Con este sistema se asegura que no se añadan transformaciones que no sean estrictamente necesarias para lograr que la serie sea estacionaria, ya que cada transformación suplementaria complica el cálculo del modelo final. Para cada transformación hay que repetir el muestreo de la serie transformada hasta que el aspecto de la serie no presente ningún tipo de tendencia o estacionalidad marcada que sugiera que la serie sigue siendo no estacionaria.
Una vez que la serie resultante sea estacionaria, el siguiente paso es proponer un modelo que se ajuste al proceso estocástico subyacente. Para ello, se aplican las funciones de autocorrelación y autocorrelación parcial sobre la serie estacionaria.

En función del comportamiento de las dos funciones, se puede determinar qué modelo es el más indicado para aproximar el proceso estocástico que se pretende simular:

Tabla 3.1. Aspectos de las funciones ACF y PACF para los modelos ARMA

	ACF	PACF
AR(p)	Disminuye sin llegar a anularse	Se anula para retardos superiores a p
MA(q)	Se anula para retardos superiores a q	Disminuye sin llegar a anularse
ARMA(p,q) con p>0 y q>0	Disminuye sin llegar a anularse	Disminuye sin llegar a anularse

Cuando se intentan modelar los procesos estocásticos de series reales, es habitual encontrar que más de un modelo puede representar con exactitud su comportamiento. En estos casos se suele escoger aquel modelo que presente un buen compromiso entre la precisión con la que ajustará al proceso y el coste computacional que significará su cálculo ya que cuanto menores sean los grados de los modelos, menor será el coste computacional del conjunto, aunque puede representar un peor ajuste a la realidad del modelo. Es en éste punto donde el analista debe juzgar qué picos son realmente significativos y representativos de la realidad del proceso y qué picos son aleatorios o accidentales.

Una vez seleccionado el tipo de modelo, sólo es necesario determinar su orden. En el caso de los modelos AR y MA, las propias funciones ACF y PACF proporcionan el mejor sistema para determinar el orden de p o q, pero en el caso de los modelos ARMA, el sistema más efectivo es la utilización de la función de autocorrelación extendida (EACF). El método de la esquina como se conoce coloquialmente, es un método gráfico que consiste en seleccionar de la gráfica de la EACF aquella intersección a partir de la cual se pueda seleccionar un triángulo de coeficientes hacia abajo y a la derecha que sólo contenga coeficientes nulos. La intersección dónde se produce esta “esquina” indicará el orden propuesto para q y para p.

3.5.2. Estimación de parámetros

Una vez determinado el modelo y su orden, el siguiente paso es la estimación del valor de los coeficientes ϕ y/o θ. Existen diversos métodos para realizar la estimación de los parámetros, como el método de los momentos y los métodos basados en mínimos cuadrados. No obstante, el método que hemos utilizado a lo largo de las pruebas realizadas, ha sido el método de Estimación por Máxima Verosimilitud (también conocido por MLE por sus siglas en ingles, *Maximum Likelihood Estimation*) [10,12].

El método de máxima verosimilitud selecciona aquellos parámetros que consiguen que la distribución que generan sea la que más probablemente haya producido los resultados observados. La estimación de máxima verosimilitud proporciona una aproximación unificada de la estimación, la cual es bien definida en el caso de una distribución normal y en gran cantidad de otros problemas. Sin embargo, en algunos problemas complejos, los estimadores de máxima verosimilitud pueden ser inadecuados o incluso no existir.

La principal ventaja de éste método es que presenta una mayor probabilidad de acercarse más al valor real de los coeficientes estimados, ya que utiliza toda la información que proporcionan los datos en lugar de sólo el primer y el segundo momento, como sucede con los métodos de mínimos cuadrados. Sin embargo, en muchos casos, las ecuaciones de máxima verosimilitud pueden ser intratables sin el uso de ordenadores mientras que el método de los momentos, por ejemplo, puede ser calculado a mano. Es precisamente esta desventaja, la que más ha retrasado la inclusión de éste método de estimación de parámetros, hasta que los avances tecnológicos han permitido una computación rápida de resultados.

3.5.3. Diagnóstico del modelo

El diagnóstico del modelo engloba dos tipos de procedimientos: los orientados a la comprobación de la eficacia de un modelo para predecir la realidad y los orientados a elegir el mejor modelo de un conjunto de candidatos.

3.5.3.1. Diagnóstico de eficacia

Uno de los sistemas más extendidos para comprobar la bondad del afinado de un modelo se basa en la información que se puede extraer de los residuales estandarizados.

Los residuales se calculan como la diferencia entre el valor observado y el valor estimado por la línea de regresión y se puede considerar como el error aleatorio observado. Los residuales estandarizados se calculan dividiendo los residuales entre la desviación estándar de los residuales y por lo tanto tienen media 0 y una desviación estándar de 1.

Idealmente, la representación gráfica de los residuales debería ser una estructura puramente aleatoria como la que presenta el ruido blanco. Esto significaría que el modelo ha conseguido captar de manera exacta el comportamiento del proceso estocástico subyacente a la serie y que las interferencias que se producen en cada predicción son solo aleatorias y no debidas a ningún tipo de correlación. No obstante, en los casos reales se suelen limitar los grados que se admiten para el modelo con el objetivo de conseguir una buena aproximación sin complicar el modelo en exceso. Este hecho provoca la aparición de picos que son el resultado de pequeñas correlaciones que no se han tratado por haber sido consideradas poco significativas en el momento de la proposición del modelo. Es en este momento cuando se puede verificar con exactitud si el analista ha acertado en el momento de no tenerlas en cuenta o se ha equivocado y son más representativas de lo que esperaba. Para tener una idea más clara de la importancia de un pico residual, es recomendable trazar el gráfico de la función de autocorrelación de los residuales estandarizados. Es en este gráfico donde se puede apreciar si efectivamente existe o no una correlación significativa para algún retraso y de ser así, ver su orden de magnitud.

3.5.3.2. Diagnóstico comparativo

En el campo de la comparación de dos modelos, se hace necesaria la obtención de una valoración o puntuación que permita determinar qué modelo presenta una mejor adecuación a la realidad. En éste ámbito se han desarrollado diversos criterios de selección, siendo el publicado por Hirotsugu Akaike en 1974 [10,13] uno de los más extensamente utilizados.

El criterio de Información de Akaike, o AIC por sus siglas en inglés (*Akaike Information Criterion*), es una medida de la bondad del ajuste de un modelo estadístico. Está basado en el concepto de entropía de la información y en efecto, ofrece una medida relativa de la información perdida cuando un modelo es usado para describir la realidad. Todos los modelos pueden ordenarse de acuerdo con su AIC, ya que el modelo que tiene un valor más pequeño de AIC es el que mejor se ajusta a la realidad. El cálculo del AIC es inmediato:

$$AIC = 2k - 2\ln(L) \qquad \textbf{(3.9)}$$

donde k es el número de parámetros en el modelo estadístico y L es el valor maximizado de la función de verosimilitud para el modelo estimado.

3.5.4. Predicción

Se puede considerar que una vez alcanzado éste punto, el análisis de series temporales, en sí mismo, ha concluido. No obstante, no tiene ningún sentido realizar éste análisis sin un objetivo que lo impulse. En nuestro caso, ese objetivo es la obtención de un modelo que permita simular el proceso estocástico subyacente a una serie real y utilizarlo para predecir su comportamiento futuro.

Para realizar una predicción con un modelo ARIMA, sólo hay que extender la misma ecuación que define al modelo hacia un retardo situado en t+l, donde l es el instante que se quiere predecir. Como se puede observar, realmente no se calcula un retardo, ya que si t es el instante considerado “actual”, el instante predicho estará a una distancia l en el futuro.

La fórmula para predecir el valor de la variable para un instante l mediante un modelo general ARIMA(p,d,q)x(P,D,Q)$_s$ es:

$$\begin{aligned}\hat{Y}(l) = \phi_1\hat{Y}_t(l-1) + \phi_2\hat{Y}_t + \phi_{l-1}Y_t(l) + \cdots + \phi_{l+1}Y_{t-l} + \cdots \\ +\phi_{Ps+p+Ds+d}Y_{t+l-Ps-p-Ds-d} + \theta_l a_t + \theta_{l+1}a_{t-1} + \cdots \\ +\theta_{Qs+q}a_{t+l-Qs-q} + \mu\end{aligned} \qquad \textbf{(3.10)}$$

Es importante destacar los siguientes aspectos de la predicción siguiendo esta formulación:

- Los valores de predicción se calculan de forma secuencial. Para hacer posible la aplicación de la parte autorregresiva de un modelo en

períodos distintos al primero de predicción, se toma el valor predicho en el inmediatamente anterior ($Y_t(l-1)$).

- Se considera que todas las perturbaciones aleatorias se conocen en el período muestral y tienen carácter de ruido blanco para los períodos de predicción.
- Por la condición necesaria para estar ante el predictor óptimo (el valor esperado de la serie en el período de predicción es igual al predicho si es óptimo), se demuestra que las perturbaciones aleatorias empleadas en la predicción son sólo las de períodos anteriores, es decir:

$$a_{t+j} \quad \forall\, j \leq 0 \tag{3.11}$$

ya que se ha supuesto que en el período de predicción, las perturbaciones aleatorias tienen esperanza nula. Para realizar las sucesivas predicciones de $Y_t(l)$ tenemos que contar con unos valores para a_t, a_{t-1}, …a_{t+l-q}. Como tales se toman:

$$a_{t+j} = Y_{t+j} - \hat{Y}_{t+j-l}(l) \quad \forall\, j \leq 0 \tag{3.12}$$

- Las predicciones ARIMA son adaptativas y los resultados obtenidos para (t+l), con la información disponible hasta el período t, son los mismos que las que se obtendrían para el mismo período tomando como base informativa hasta t-1, y añadiendo un término de error.

CAPÍTULO 4. ANÁLISIS ARIMA SOBRE REDIRIS

4.1. Introducción

La topología de RedIRIS está constituida por 33 enlaces bidireccionales, lo que implica la existencia de 66 series temporales asociadas a distintos flujos de datos. Para mostrar una aplicación práctica del método ARIMA hemos elegido describir aquí el enlace and-can-l por ser uno de los más representativos. El resto de análisis efectuados se pueden encontrar en el anexo A.III.

4.2. El análisis de la serie del enlace and-can-l

La serie and-can-l es la serie que representa los flujos de datos que atravesaron el enlace que une el router de Andalucía con el de Las Palmas de Gran Canaria para los días contenidos en los registros. Se trata de uno de los enlaces primarios que presentan una mejor disposición a ofrecer una predicción válida.

Todos los cálculos que conforman el análisis se han realizado mediante el entorno de análisis estadístico R [10,14].

4.2.1. Especificación de modelos

Una vez introducida la serie temporal en el entorno de R, recordemos que el primer paso para realizar un análisis de series temporales es la representación gráfica, ya que permite el descubrimiento de tendencias o estacionalidades que pongan de manifiesto la no-estacionariedad de la serie:

```
plot(and_can_l)
```

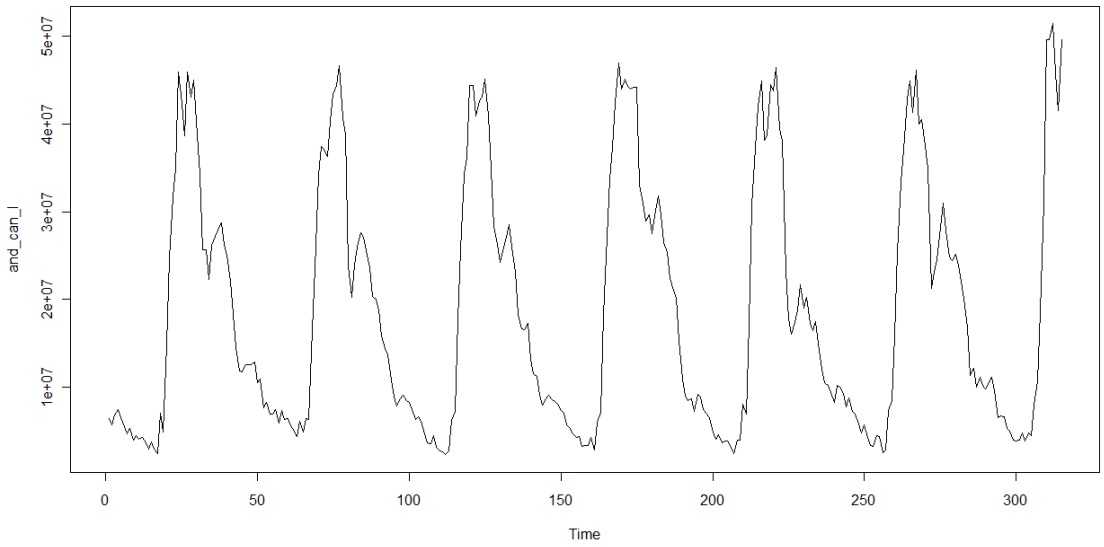


Fig. 4.1 Gráfica de la serie and-can-l

La inspección visual de la gráfica indica que la serie presenta una clara estacionalidad.

Como se ha indicado en el capítulo de la teoría del análisis ARIMA, la función de autocorrelación se puede utilizar para poner de manifiesto patrones repetitivos como la periodicidad de una estacionalidad:

```
acf(as.vector(and_can_l),lag.max=144)
```

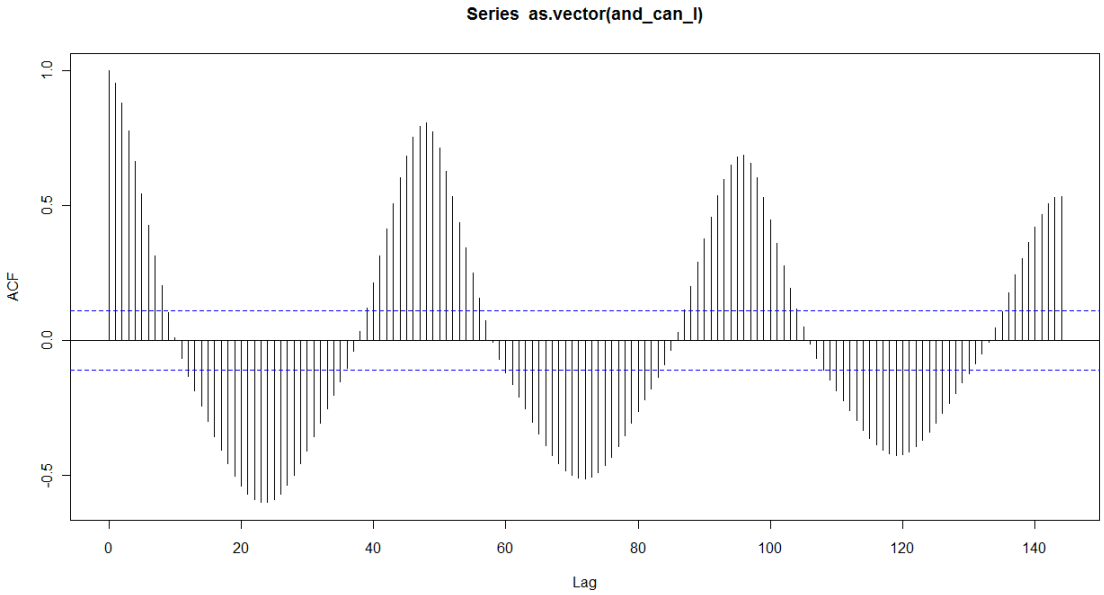


Fig. 4.2 Gráfica de la función de autocorrelación

Se puede observar que la función de autocorrelación es especialmente significativa para los retrasos múltiplos de 48. Como cada registro contiene la

media de los datos transferidos durante 1800 segundos, el patrón se repite cada 86400 segundos, es decir, cada día. Así, la estacionalidad de la serie queda definida a un período de 48 muestras, una estacionalidad diaria. El siguiente paso consistirá en eliminar la estacionalidad detectada y para ello se llevará a cabo una diferenciación para el retraso 48.

Con el objetivo de realizar una comparativa de dos modelos propuestos para una misma serie temporal, se definirán dos rutas de acción para la obtención de dos modelos ligeramente diferentes. En la primera, sólo derivaremos estacionalmente y en la segunda, además de derivar estacionalmente, aplicaremos la primera derivada sobre el modelo. Partiendo de la necesidad de eliminar la estacionalidad a 48, ¿por qué aplicar también la primera derivada al modelo? Este segundo modelo tiene como objetivo comprobar si hay una no estacionariedad que no se ha observado una vez aplicada la derivada estacional.

Tomando estas dos rutas más probables, esperamos conseguir 2 modelos bastante adecuados, eligiendo después el más prometedor, que será aquel que presente el mejor compromiso entre simplicidad y adecuación.

Para determinar la cantidad de órdenes contemplados para cada modelo, se seguirá el siguiente criterio: las correlaciones que se considerarán significativas serán aquellas que superen un factor de ±0,2 y con no más de 4 retrasos, que superen los umbrales de significancia del 5%, admitiendo el retraso 48 como necesario a nivel estacional.

4.2.1.1. Especificación del modelo para la primera ruta de acción

Primero, se deriva estacionalmente para el retraso 48:

```
plot(diff(and_can_l,lag=48))
```

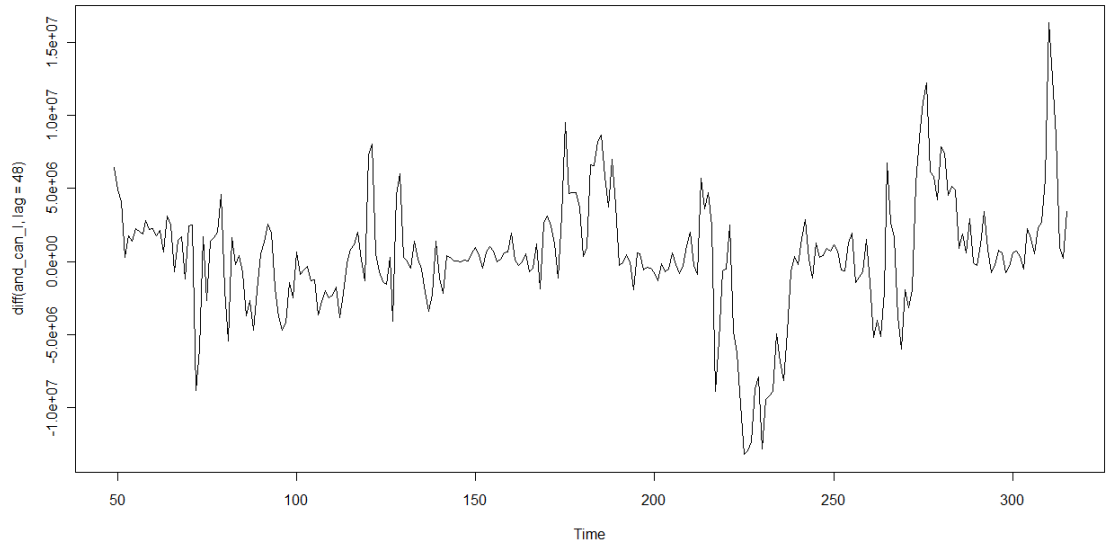


Fig. 4.3 Gráfica de la serie and-can-l diferenciada para el retraso 48

Como se puede observar, la serie transformada ya no presenta la estacionalidad. A continuación, se procede a establecer qué modelo ARIMA es el más adecuado para representar a la serie transformada.

Aplicando el ACF sobre esta transformación:

```
acf(as.vector(diff((and_can_l),lag=48)),lag.max=144,ci.type='ma')
```

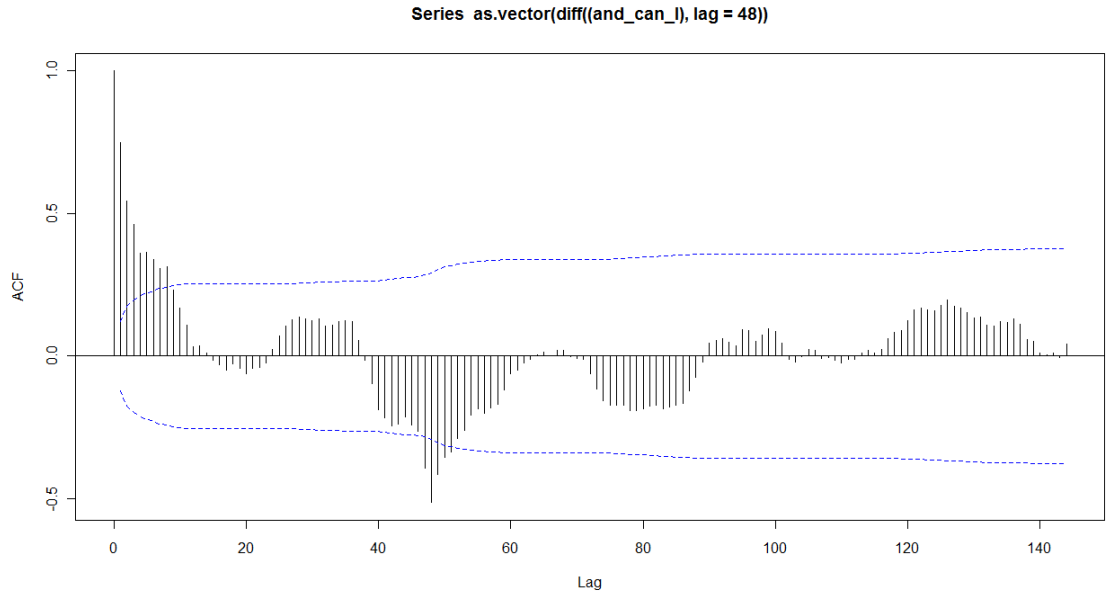


Fig. 4.4 Gráfica de la función de autocorrelación

Aplicando el PACF sobre la transformación:

```
pacf(as.vector(diff((and_can_l),lag=48)),lag.max=144)
```

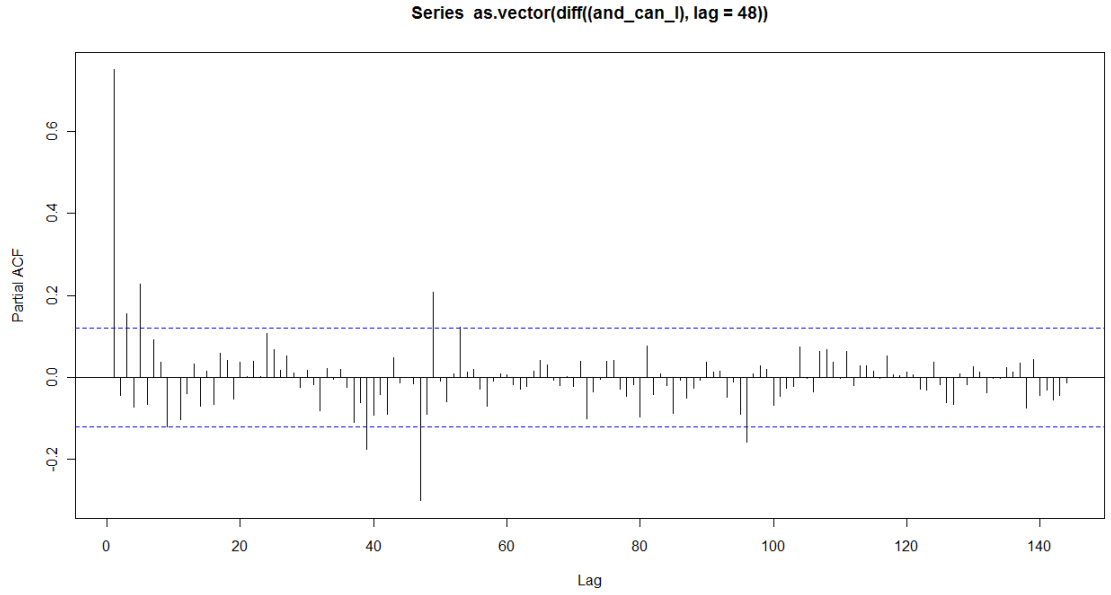


Fig. 4.5 Gráfica de la función de autocorrelación parcial

Observando estas funciones y comparándolas con los comportamientos asociados a cada modelo ARMA, se puede sugerir que un modelo que incorpore los retrasos autorregresivos 1 y 48 sería adecuado.

El modelo sugerido para la primera ruta es un modelo multiplicativo estacional ARIMA $(1,0,0)x(1,1,0)_{48}$

4.2.1.2. Especificación del modelo para la segunda ruta de acción

Primero, se deriva estacionalmente para el retraso 48 y sobre el resultado, se aplica la primera derivada:

```
plot(diff(diff((and_can_l),lag=48)))
```

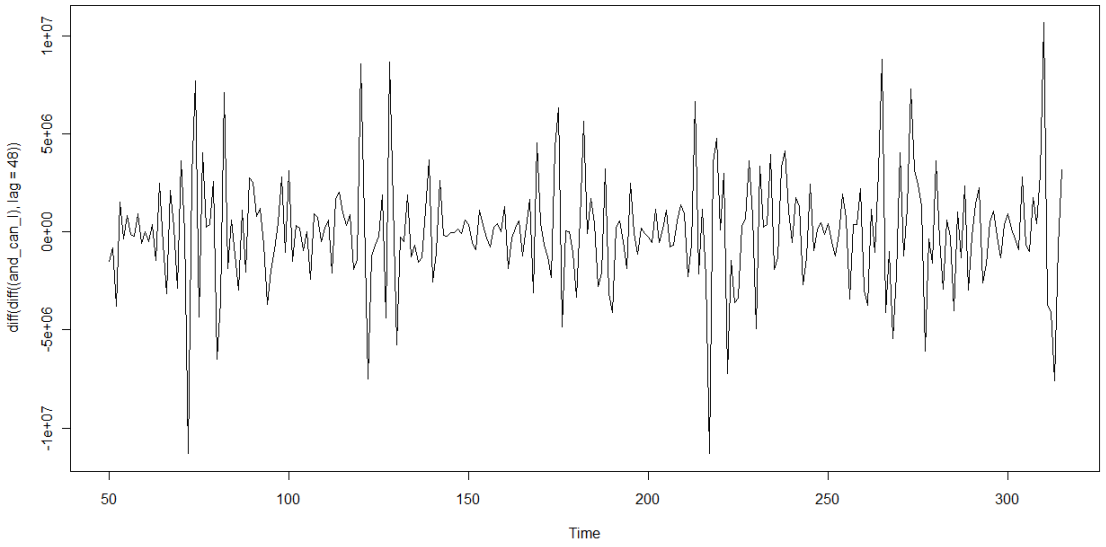


Fig. 4.6 Gráfica de la serie and-can-l diferenciada para los retrasos 1 y 48

La serie transformada ya no presenta la estacionalidad ni tampoco puede presentar una tendencia. A continuación, se procede a establecer qué modelo ARIMA es el más propicio para representar a la serie transformada.

Aplicando el ACF sobre esta transformación:

```
acf(as.vector(diff(diff((and_can_l),lag=48))),lag.max=144,ci.type='ma')
```

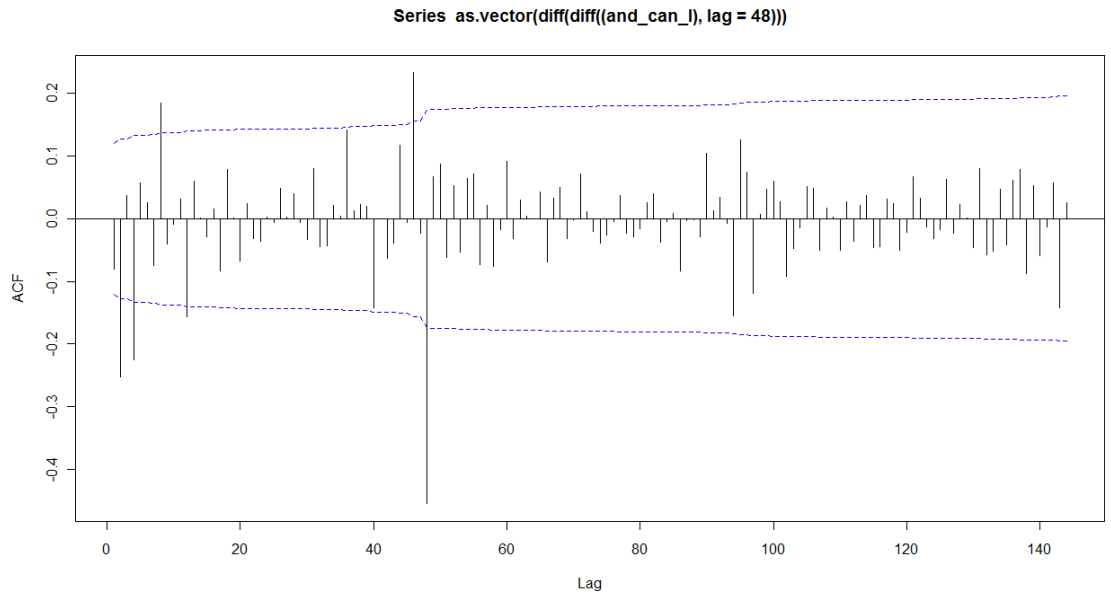


Fig. 4.7 Gráfica de la función de autocorrelación

Aplicando el PACF sobre la transformación:

```
pacf(as.vector(diff(diff((and_can_l),lag=48))),lag.max=144)
```

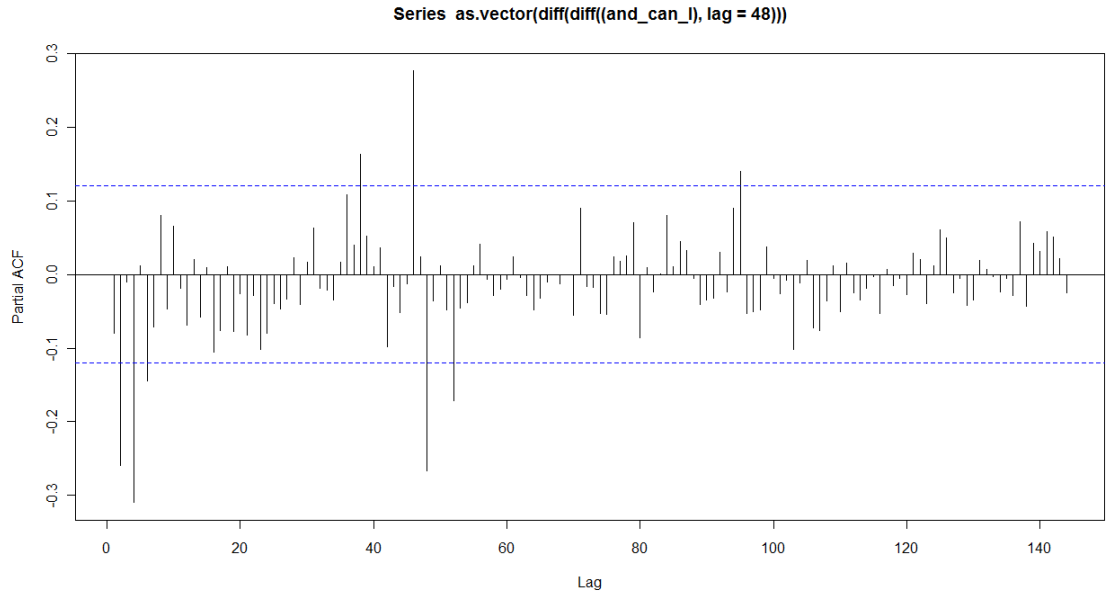


Fig. 4.8 Gráfica de la función de autocorrelación parcial

Si se vuelve a observar la gráfica del ACF, se puede sugerir que un modelo de medias móviles que incorpore los retrasos 4 y 48 sería adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(0,1,4)x(0,1,1)_{48}$

4.2.2. Estimación de parámetros

Mediante el uso del MLE, se realiza la estimación de los parámetros ϕ y θ para los modelos propuestos.

4.2.2.1. Estimación de parámetros para el modelo ARIMA(1,0,0)x(1,1,0)$_{48}$

```
m1.and_can_l=arima(and_can_l,order=c(1,0,0),seasonal=list(order=c(1,1,
0),period=48))
m1.and_can_l
```

```
Call:
arima(x = and_can_l, order = c(1, 0, 0), seasonal = list(order = c(1,
1, 0),
    period = 48))

Coefficients:
         ar1     sar1
      0.7339  -0.6183
s.e.  0.0417   0.0506

sigma^2 estimated as 4.513e+12:  log likelihood = -4280.73,  aic =
8567.46
```

Fig. 4.9 Resultados del MLE para el modelo ARIMA(1,0,0)x(1,1,0)$_{48}$

4.2.2.2. Estimación de parámetros para el modelo ARIMA(0,1,4)x(0,1,1)$_{48}$

```
m2.and_can_l=arima(and_can_l,order=c(0,1,4),seasonal=list(order=c(0,1,
1),period=48))
m2.and_can_l
```

```
Call:
arima(x = and_can_l, order = c(0, 1, 4), seasonal = list(order = c(0,
1, 1),
    period = 48))

Coefficients:
          ma1      ma2      ma3     ma4     sma1
      -0.1685  -0.2928  -0.0803  -0.194  -0.9999
s.e.   0.0605   0.0619   0.0612   0.062   0.2085

sigma^2 estimated as 3.190e+12:  log likelihood = -4251.98,  aic =
8513.95
```

Fig. 4.10 Resultados del MLE para el modelo ARIMA(0,1,4)x(0,1,1)$_{48}$

4.2.3. Diagnóstico del modelo

A continuación procederemos a comprobar la bondad del afinado individual de los modelos y para ello, primero hay que observar los residuales estandarizados.

4.2.3.1. Diagnóstico de eficacia del modelo ARIMA(1,0,0)x(1,1,0)$_{48}$

```
plot(window(rstandard(m1.and_can_l)),ylab='Residuales Estandarizados',
type='o')
abline(h=0)
```

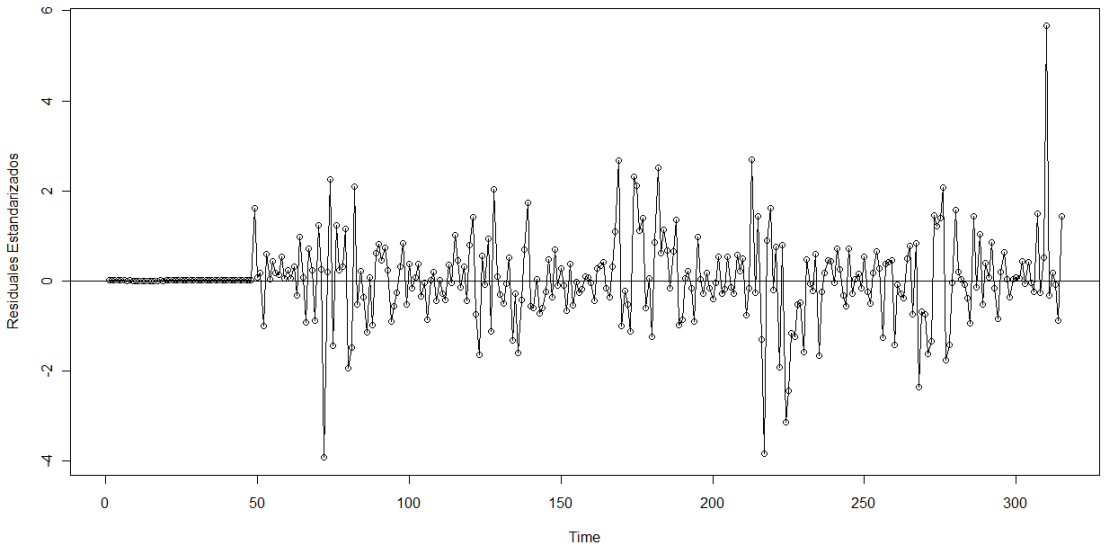


Fig. 4.11 Gráfica de los residuales estandarizados

Salvo por unos tres picos a mediados de la serie y otro al final, el resto de residuales se encuentran acotados entre 2 y -2 aproximadamente. Para precisar más, visualizaremos el ACF de los residuales.

```
acf(as.vector(window(rstandard(m1.and_can_l))),lag.max=144)
```

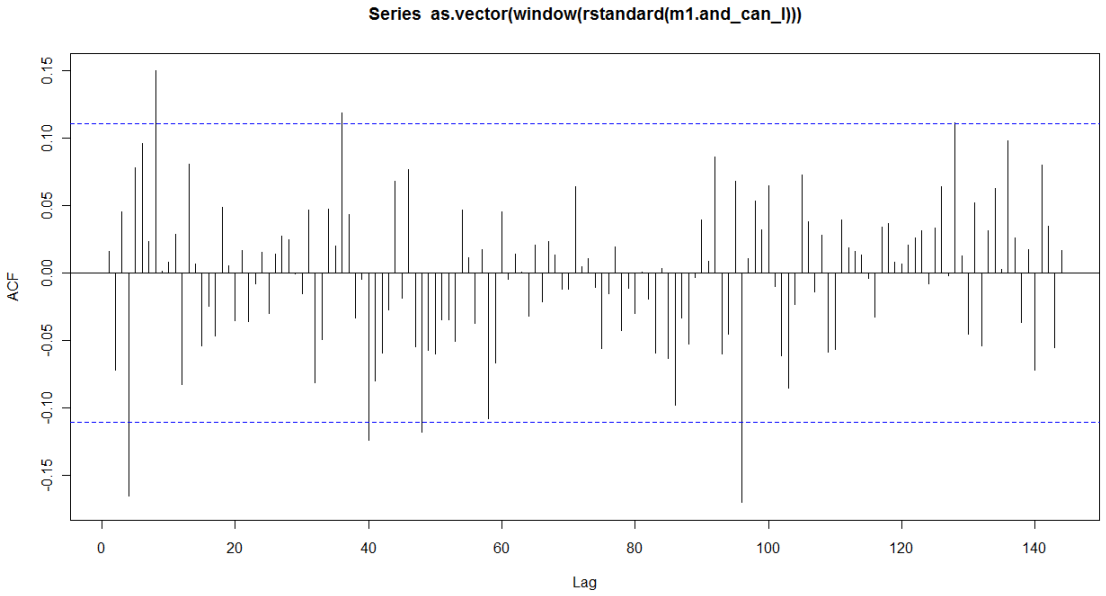


Fig. 4.12 Gráfica del ACF de los residuales estandarizados

Se aprecian seis correlaciones “estadísticamente significativas”. La correlación más significativa presenta un valor de correlación del entorno del -0,17, una correlación muy pequeña. Además, no tenemos una clara razón para pensar que los retrasos 4 y 8 se deban a algo más que a la casualidad. No obstante, el retraso 96 sí que podría ser indicativo de algo más, ya que es múltiplo de la frecuencia de datos, 48. Aun así, se puede decir que el modelo ha capturado la esencia del proceso estocástico subyacente a la serie.

Por otra parte, se puede calcular la relación señal-ruido SNR [15] como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

```
var(and_can_l)
```

```
[1] 2.006534e+14
```

SNR = 2.006534e+14 / 4.513e+12 = 44,46

4.2.3.2. Diagnóstico de eficacia del modelo ARIMA(0,1,4)x(0,1,1)$_{48}$

```
plot(window(rstandard(m2.and_can_l)),ylab='Residuales Estandarizados',
type='o')
abline(h=0)
```

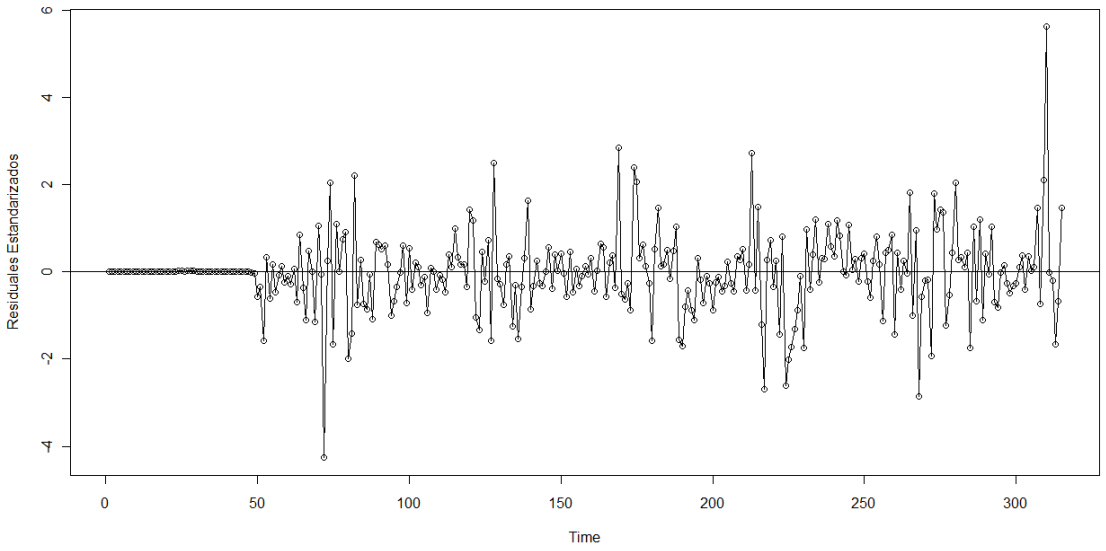


Fig. 4.13 Gráfica de los residuales estandarizados

Presenta casi la misma distribución que para el modelo anterior, salvo que en este caso, dos de los residuales significativos han sido reducidos. Para precisar más, visualizaremos el ACF de los residuales.

```
acf(as.vector(window(rstandard(m2.and_can_l))),lag.max=144)
```

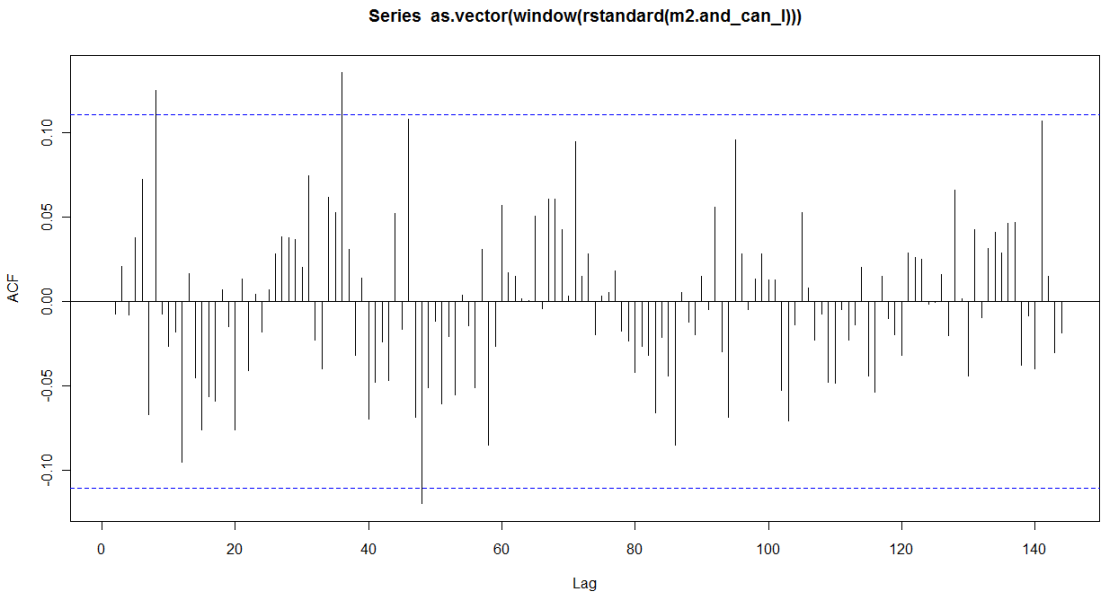


Fig. 4.14 Gráfica del ACF de los residuales estandarizados

Decididamente, las correlaciones significativas se han visto reducidas en número (tres en lugar de seis) y en fuerza, con la correlación más significativa a un valor de cerca de 0,14.

Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

```
var(and_can_l)
```

```
[1] 2.006534e+14
```

SNR = 2.006534e+14 / 3.190e+12 = 62,9

Una vez concluido el diagnóstico de eficacia individual de cada modelo, es el momento de comparar ambos modelos y seleccionar aquel que presenta una mejor adecuación a la serie real, basándonos en criterios como el AIC.

4.2.3.3. Diagnóstico comparativo

Tabla 4.1. Comparativa del AIC y el SNR de ambos modelos

	AIC	SNR
ARIMA(1,0,0)x(1,1,0)$_{48}$	8567.46	44,46
ARIMA(0,1,4)x(0,1,1)$_{48}$	8513.95	62,9

Si comparamos los resultados de ambos modelos, el modelo propuesto como segunda opción ARIMA(0,1,4)x(0,1,1)$_{48}$, presenta una mejor relación señal-ruido y unas correlaciones de los residuales más bajas, lo que indica una mejor adecuación a la serie temporal. Esto se corresponde con lo indicado por el AIC, ya que el valor menor lo presenta éste modelo.

Finalmente, el último paso a realizar es la predicción de algún resultado futuro para el modelo escogido.

4.2.4. Predicción

Para concluir este análisis, hemos realizado una predicción del comportamiento futuro de la serie and-can-l para los siguientes 3 días, según el modelo ARIMA(0,1,4)x(0,1,1)$_{48}$.

```
plot(m2.and_can_l,n.ahead=144,xlab='Time',type='o',ylab='Trafico and-can-l')
abline(h=0)
```

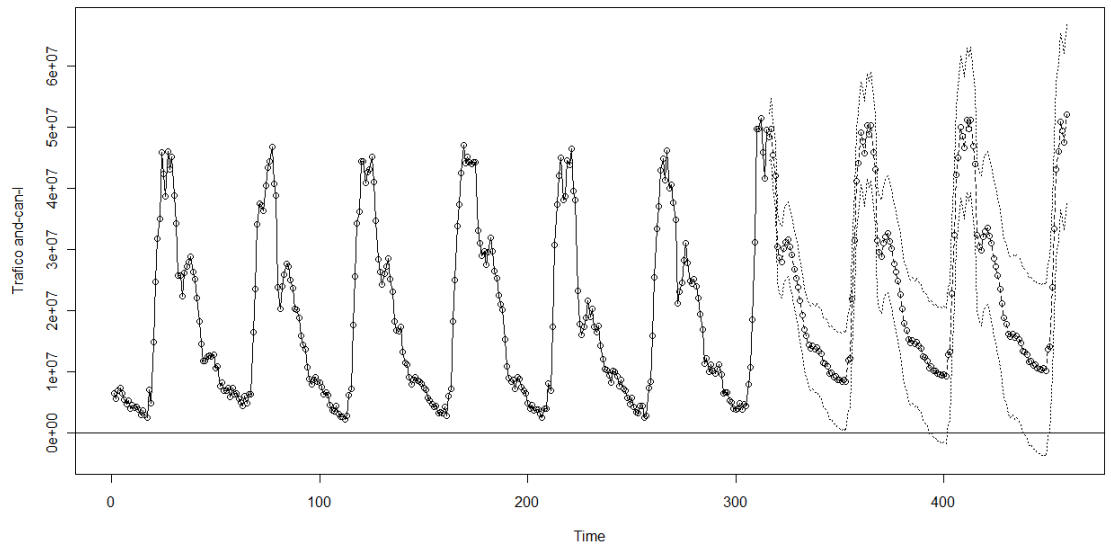


Fig. 4.15 Gráfica de la serie and-can-l y su predicción para los 3 días siguientes

Las líneas discontinuas marcan el intervalo de variación máximo estimado para la serie. Es importante destacar que aunque el programa estima la posibilidad de la aparición de valores inferiores a cero sabemos, por la naturaleza de los datos, que es imposible ya que cero representa la no utilización de un enlace. Para facilitar la observación de este hecho, se ha añadido una línea en cero.

También se puede observar la progresiva degeneración de la predicción, representada mediante la abertura cada vez mayor del intervalo de variación máximo. Este hecho se debe a que, a medida que nos alejamos de los valores conocidos, las predicciones se generan a partir de las predicciones inmediatamente anteriores. Esto implica que, al final, el contacto con la realidad se puede haber perdido y los resultados dejan de tener valor.

A continuación realizamos una representación gráfica del afinado de la predicción para los dos modelos especificados en el análisis anterior:

```
plot(m1.jj,n.ahead=336,xlab='Time',type='o',ylab='Trafico and-can-l')
```

```
lines(jk,col="blue")
```

ARIMA $(1,0,0)x(1,1,0)_{48}$

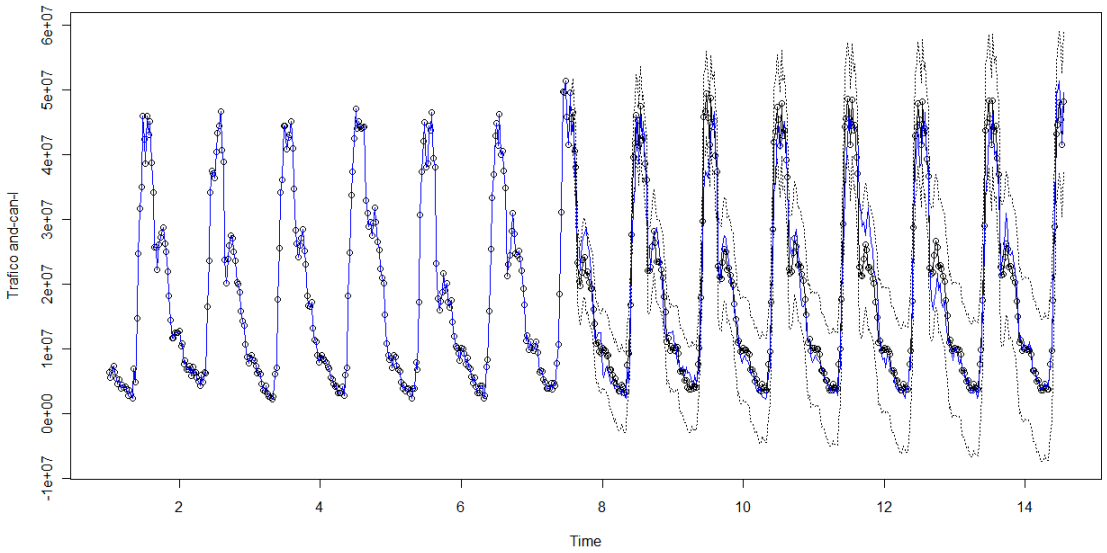


Fig. 4.16 Afinado de la predicción con ARIMA $(1,0,0)x(1,1,0)_{48}$

ARIMA $(0,1,4)x(0,1,1)_{48}$

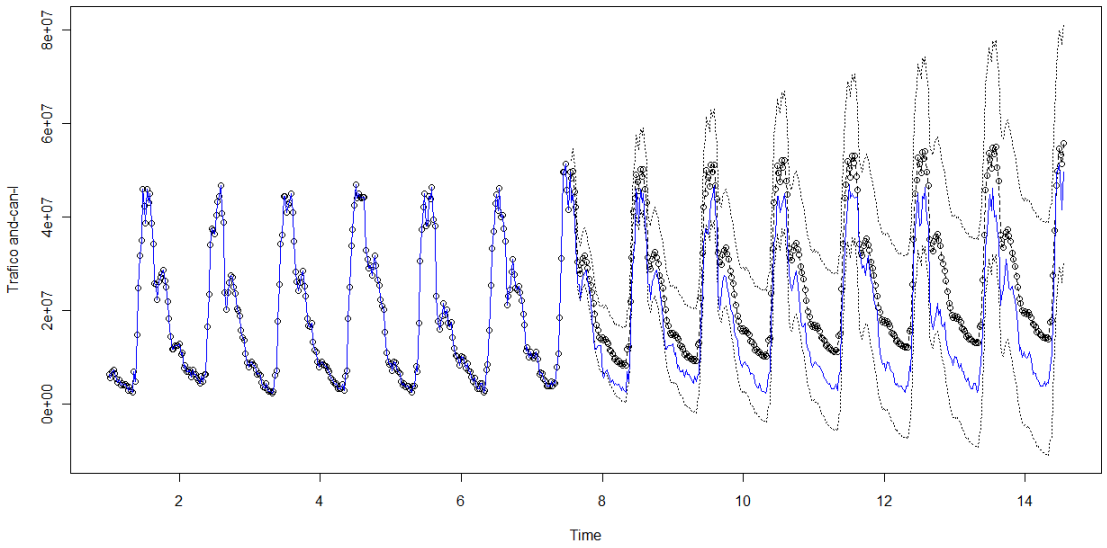


Fig. 4.17 Afinado de la predicción con ARIMA $(0,1,4)x(0,1,1)_{48}$

Como se puede comprobar en ambas gráficas, el modelo que prometía ser el que mejor ajustaba al proceso subyacente de la serie es el que presenta un peor ajuste predictivo con la realidad. Esta situación es normal y viene condicionada por la variabilidad intrínseca que nunca va a poder capturar el modelo como periodicidades ocultas, procesos intrínsecamente caóticos, etc.

CAPÍTULO 5. ANÁLISIS DE COMPONENTES PRINCIPALES

5.1. Introducción

El análisis de componentes principales (en español ACP, PCA en inglés) es una técnica cuyo objetivo es la reducción de las dimensiones (o número de variables) de un conjunto de datos, perdiendo por ello la mínima cantidad de información posible [16, 17, 18]. Es utilizada principalmente como herramienta en el análisis exploratorio de datos y para la realización de modelos predictivos. Fue introducida en 1901 por Karl Pearson y desarrollada independientemente en 1933 por Hotelling. Su primera implementación computacional fue realizada por Jean Pages en los años 60 del pasado siglo para analizar encuestas de opinión pública. Dependiendo del campo de aplicación, también se conoce como la Transformación Discreta de Karhunen-Loève, la Transformación de Hotelling o la Descomposición Ortogonal Propia [16].

5.2. Descripción del análisis

La reducción de dimensiones que caracteriza a este análisis se lleva a cabo mediante un proceso matemático que transforma un conjunto de variables correlacionadas en un conjunto de variables no relacionadas entre ellas llamadas componentes principales que concentran la mayor cantidad posible de información:

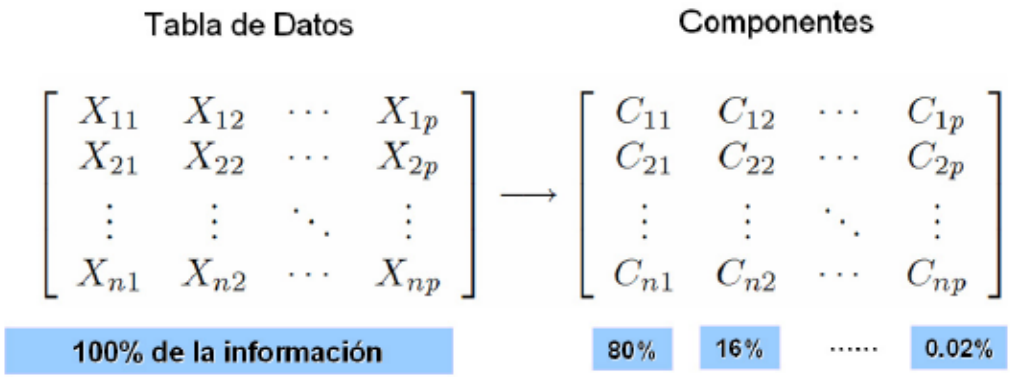


Fig. 5.1 Transformación de las variables originales en componentes principales

Los componentes principales son calculados como una combinación lineal de las variables originales y tienen la propiedad de ser linealmente independientes. Así pues, las nuevas componentes principales están relacionadas con las variables originales mediante una transformación lineal.

5.2.1. La idea subyacente

De manera intuitiva, los *n* datos de una matriz se pueden interpretar como una nube de puntos en R^p. La idea subyacente en la transformación es buscar una

proyección que capture la máxima varianza de la nube de puntos, substituyendo si es necesario, los ejes habituales por otros que sean más adecuados para el conjunto de puntos estudiado. Si se realiza una proyección de los puntos sobre un nuevo subespacio como puede ser un plano, de manera que éste capture la máxima varianza de los puntos, se puede utilizar como base sobre la que trazar unos nuevos ejes que se adecuen mejor a la distribución particular de los puntos. Además, la proyección de los puntos sobre un plano representa una reducción de dimensiones en sí misma, ya que se pueden ignorar terceras dimensiones y contemplar los puntos sobre sólo dos dimensiones.

Si observamos el caso particular de un conjunto de datos representados sobre R^3, la nube de puntos no suele presentar un patrón de distribución claro como se observa en el ejemplo de la figura a). Sobre esta representación se puede buscar un plano que recoja la máxima varianza de la nube de puntos, como se muestra en la figura b). Si a continuación obviamos la tercera dimensión y reducimos la observación al subespacio de R^2 que traza el plano, como se ve en la figura c), habremos llevado a cabo una reducción de las dimensiones del sistema a la par que disponemos de una representación simplificada del sistema sobre la que observar relaciones entre puntos que podrían haber pasado inadvertidas en la representación canónica.

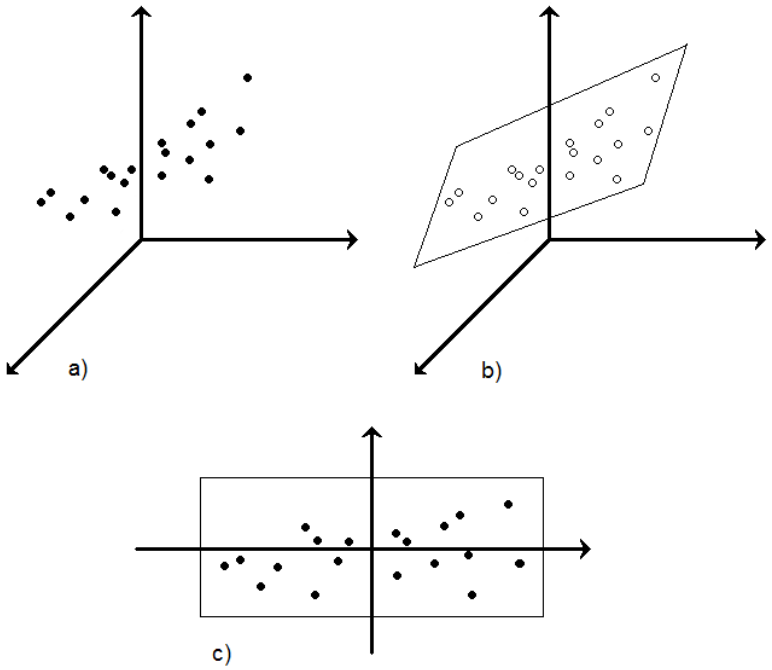


Fig. 5.2 Proyección de los datos en el plano de varianza máxima

5.2.2. La definición estricta

Concretamente, la transformación se construye escogiendo un nuevo sistema de coordenadas para el conjunto original de datos en el cual la varianza de

mayor tamaño es capturada en el primer eje (llamado el Primer Componente Principal), la segunda varianza más grande es el segundo eje y así sucesivamente, bajo la condición de que cada nuevo componente sea ortogonal a los componentes precedentes, añadiendo así la propiedad de ortogonalidad a la transformación lineal.

Estos nuevos componentes principales se pueden utilizar para construir modelos genéricos sencillos que permitan predecir comportamientos futuros o inferir series que no se habían utilizado para el análisis pero que comparten una fuerte correlación con éstas. Por esta razón, el objeto de estudio del PCA suelen ser matrices compuestas por series de datos que recogen un mismo conjunto de variables medidas para un único individuo o entidad. Estas variables pueden ser multidisciplinarias (como las respuestas de una encuesta) o por el contrario representar la evolución temporal de una sola variable.

El método de extracción de las componentes principales se compone de los siguientes cuatro pasos:

1. Si X es un vector p-dimensional con matriz de covarianza Σ, el primer componente principal es una combinación lineal $X \cdot g$ para algún vector g que satisface $g^T \cdot g = 1$. El vector g se escoge para maximizar $g^T \cdot \Sigma \cdot g$ sujeto a la condición $g^T \cdot g = 1$. Si la solución del problema es g_1, entonces $X \cdot g_1$ es el primer componente principal. Hay que señalar que si $z_1 = X \cdot g_1$ entonces, $\text{var}(z_1) = \text{var}(X \cdot g_1) = g_1^T \cdot \Sigma \cdot g_1$.
2. Para obtener otro componente principal, se repite el proceso de optimización en el espacio ortogonal a g_1 y se encuentra un g_2 que maximice $g_2^T \cdot \Sigma \cdot g_2$, tal que $g_2^T \cdot g_2 = 1$ y $g_2^T \cdot g_1 = 0$. Entonces $X \cdot g_2$ es el segundo componente principal.
3. En general, sea $g_1,\ldots,g_{k-1}$ para algún $k \leq p$, entonces encontramos g_k que maximice $g_k^T \cdot \Sigma \cdot g_k$ sujeto a $g_k^T \cdot g_k = 1$ y $g_k^T \cdot g_j = 0$ para $j = 1,\ldots,k\text{-}1$. Entonces $X \cdot g_k$ es el k-ésimo componente principal.
4. El proceso continua iterativamente hasta encontrar todos los p componentes.

Aunque en la práctica no se suele conocer Σ y hay estimarla a través de la matriz de covarianza muestral S, el procedimiento a seguir es el mismo que para Σ.

5.3. Propiedades del método

El proceso de extracción de los componentes principales cumple las siguientes cuatro propiedades:

- Sea G una matriz ortogonal tal que $G^T \cdot \Sigma \cdot G = D$, donde D es una matriz diagonal con entradas de la diagonal ordenadas tal que $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_p \geq 0$ y sea g_k la k-ésima columna de G, se tiene que g_k es el autovector de

norma 1 de Σ con autovalor λ_k. Siempre se puede encontrar esta representación porque Σ es una matriz simétrica y definida no-negativa.

- Los componentes principales $X \cdot g_1, X \cdot g_2, \ldots, X \cdot g_p$ no están correlacionados y la suma de sus varianzas es la suma de las varianzas de los componentes individuales de *X*.
- El problema de optimización se reduce a:

$$max_g(g^T \Sigma g) \quad \textbf{(5.1)}$$

Siempre sujeto a la condición $g^T \cdot g = 1$. El lagrangiano del problema es:

$$t = g^T \Sigma g - \lambda(g^T g - 1) \quad \textbf{(5.2)}$$

Al derivar t con respecto a g se tiene:

$$\frac{\delta t}{\delta g} = 2\Sigma g - 2\lambda g = 0 \quad \textbf{(5.3)}$$

O también:

$$(\Sigma - \lambda I)g = 0 \quad \textbf{(5.4)}$$

Dado que g es distinto de cero, el sistema anterior tiene solución única si:

$$|\Sigma - \lambda I| = 0 \quad \textbf{(5.5)}$$

Todo esto implica que λ es un autovalor de la matriz Σ y g es el autovector correspondiente.
También hay que destacar que:

$$\Sigma g = \lambda g \quad \textbf{(5.6)}$$

Y al premultiplicar por g^T se tiene:

$$g^T \Sigma g = \lambda g^T g = \lambda = var(z_1) \quad \textbf{(5.7)}$$

Lo que implica que λ corresponde al autovalor que maximiza la varianza del primer componente y es el de mayor valor posible.

- La suma de las varianzas de los componentes principales es la suma de las varianzas de las columnas de *X*:

$$tr(\Sigma) = tr(G^T \Sigma G) = tr(D) = \sum_k \lambda_k \quad \textbf{(5.8)}$$

Es importante señalar que encontrar los componentes principales no es invariante a las escalas de medición, o lo que es lo mismo, si se cambian las unidades de medición de los componentes de *X*, los componentes principales y los λ's cambian. Para evitar este problema, se puede substituir la matriz de correlaciones por la de covarianzas con lo que de hecho, existen dos submétodos para extraer los componentes principales:

1. El método basado en la matriz de correlaciones, que se aplica cuando los datos no son dimensionalmente homogéneos o el orden de magnitud de las variables aleatorias medidas no es el mismo.
2. El método basado en la matriz de covarianzas, que se aplica cuando los datos son dimensionalmente homogéneos y presentan valores medios similares.

Salvo esta diferencia, el proceso es el mismo para ambos métodos. No hace falta estandarizar las variables originales si estas están en la misma escala.

5.4. La reducción de las dimensiones

Una vez obtenidos todos los componentes principales, se puede proceder a la reducción de las dimensiones. El resultado del proceso de extracción es la obtención de los p componentes principales que se corresponden con las p dimensiones que caracterizan al modelo. Estos están ordenados por orden de importancia y tienen una cantidad menguante de información contenida en cada uno de ellos. En concreto, la proporción de la varianza explicada por los primeros k componentes se puede calcular como:

$$\phi_k = \frac{\lambda_1 + \cdots + \lambda_k}{\lambda_1 + \cdots + \lambda_p} \quad \textbf{(5.9)}$$

El objetivo es establecer un compromiso entre la cantidad de componentes principales que se desean preservar y la cantidad de información perdida como consecuencia de ello. Para ello, se suele escoger una *k* tal que ϕ_k esté cerca de 1. Los métodos propuestos para ello son:

1. Gráfico de los λ_k vs. *k* o screeplot: el método consiste en observar a partir de qué valor de *k* la gráfica se aplana.
2. Umbral de confianza: consiste en escoger un valor de *k* tal que $\phi_k \geq c$ para algún punto de corte arbitrario c, donde c es el porcentaje de información que se va a preservar. Los valores usuales para c son 0.9, 0.95 o 0.99.
3. La Regla de Kaiser: propuesto por Kaiser [10] en 1960 consiste en excluir todos los componentes principales con autovalores menores que el promedio de todos ellos. En esencia equivale a decir que, a menos que un factor extraiga al menos tanta información como el equivalente de una variable original, podemos desecharlo. Si se usa la matriz de correlación, el promedio es 1.

5.5. Predicción

Una vez escogidos los componentes principales que cumplen el criterio de selección, se puede proceder a predecir algún resultado. Para ello, es común realizar una regresión lineal con los datos facilitados por los componentes principales. Las predicciones mediante el uso de una regresión lineal pueden ser hacia el futuro o paralelas a los datos utilizados (inferencia de series).

CAPÍTULO 6. PCA APLICADO A REDIRIS

6.1. Introducción

Este capítulo se divide en dos partes. En la primera, se realizará un análisis de componentes principales (PCA) limitado a un grupo de seis series temporales especialmente seleccionadas por compartir un marcado patrón estacional. En esta fase, se pretende ilustrar la efectividad del método para generar un modelo único y eficaz, capaz de pronosticar con fidelidad el comportamiento de todas las series analizadas cuando éstas comparten un alto parecido. También se pretende demostrar la capacidad para extrapolar resultados de series no utilizadas en el análisis pero que comparten el mismo patrón.

En la segunda fase el PCA se realizará sobre el conjunto total de los datos de los enlaces que componen RedIRIS. En esta fase se pretende demostrar la eficacia del modelo para detectar los patrones de conducta más generales (como los patrones estacionales) y demostrar también su adecuación, más limitada que en el caso de la fase anterior, pero aún así correcta, para pronosticar un número elevado de series mediante un único modelo genérico. También se busca demostrar la capacidad de comprimir la representación de las series en unos pocos parámetros [19,20]. En esta fase también se extrapolarán resultados de series no utilizadas en el análisis con el objetivo de comprobar si sus pronósticos pueden ser aplicables.

6.2. PCA limitado a seis enlaces primarios

Las seis series que se utilizarán en este análisis son las correspondientes a los enlaces: and-can-l, gal-ast, mad-nac1, nac-and, nac-clm y nac-gal. También se utilizará la serie del enlace nac-ext, aunque su utilización quedará limitada sólo a cálculos comparativos. Tal como se hizo en el capítulo 4, R será el entorno en el que se realizarán todos los cálculos [21].

6.2.1. Cálculo del PCA

Una vez introducidas todas las series temporales en el entorno de R, hay que combinarlas en una misma matriz de datos:

```
fin_opt<-cbind(and_can_l,gal_ast,mad_nac1,nac_and,nac_clm, nac_gal)
```

Es importante señalar que para que el análisis se lleve a cabo con éxito en el entorno de R, la distribución correcta de esta matriz de datos ha de ser la de una matriz de dimensiones MxN, donde M son las muestras para cada serie y N son las series utilizadas para el análisis.

El siguiente paso es la realización del PCA. En este caso se ha escogido el método que lo calcula mediante la matriz de correlación:

```
pca <- prcomp(fin_opt, scale = TRUE)
pca
```

```
Standard deviations:
[1] 2.3675082 0.4290433 0.3143731 0.2179033 0.1880343
[6] 0.1707564

Rotation:
                PC1           PC2          PC3           PC4
and_can_l 0.4060555  0.184512075  0.80667957  0.066597125
gal_ast   0.4107067  0.226523315 -0.42978988  0.713248744
mad_nac1  0.3895117 -0.895588255  0.05864424  0.105446762
nac_and   0.4128369  0.315318100  0.11914477 -0.007801179
nac_clm   0.4137334  0.114330960 -0.30268304 -0.658413760
nac_gal   0.4160648  0.008190667 -0.23515332 -0.205310729
                PC5         PC6
and_can_l -0.3551573 -0.14060713
gal_ast   -0.1565866 -0.24908778
mad_nac1   0.1620649 -0.07334226
nac_and    0.8150295  0.22717980
nac_clm   -0.0667439 -0.53495110
nac_gal   -0.3928752  0.75830237
```

Fig. 6.1 Datos generales del PCA

Y más concretamente, para obtener el porcentaje de varianza explicado por cada componente principal:

```
summary(pca)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6
Standard deviation     2.368 0.4290 0.3144 0.21790 0.18803 0.17076
Proportion of Variance 0.934 0.0307 0.0165 0.00791 0.00589 0.00486
Cumulative Proportion  0.934 0.9649 0.9813 0.98925 0.99514 1.00000
```

Fig. 6.2 Datos de la varianza para cada componente principal

A continuación, se aplican los criterios de selección para determinar cuántos componentes principales se van a utilizar para generar el modelo regresivo. Según el método escogido [22], estos son:

- Kaiser: sólo hay que mantener el primer componente principal, ya que es el único que cumple estrictamente que su desviación estándar es superior o igual a 1.
- 90% de la varianza: como en el método de káiser, sólo se ha de mantener el primer componente principal.
- 95% de la varianza: se han de mantener los dos primeros componentes principales.
- 99% de la varianza: se ha de mantener hasta el quinto componente principal.
- 100% de la varianza: los seis componentes principales.

A medida que se mantienen porcentajes de varianza más elevados, se incluyen más componentes principales y la serie es más fiel a la original, a costa de un número superior de parámetros y una representación que puede acabar resultando demasiado específica.

Si se almacenan las acumulaciones de componentes principales en variables, resulta mucho más claro manejar las instrucciones de R:

```
kaiser<-pca$x[,1]
var95<-pca$x[,1]+pca$x[,2]
var99<-pca$x[,1]+pca$x[,2]+pca$x[,3]+pca$x[,4]+pca$x[,5]
var100<-pca$x[,1]+pca$x[,2]+pca$x[,3]+pca$x[,4]+pca$x[,5]+ pca$x[,6]
```

Se pueden obtener gráficamente los resultados del análisis para cada criterio de selección. Para facilitar su comparación, los dibujaremos solapados, asignando un color a cada criterio:

```
plot(kaiser,type='l',col="black")
lines(var95,col="blue")
lines(var99,col="red")
lines(var100,col="darkgreen")
```

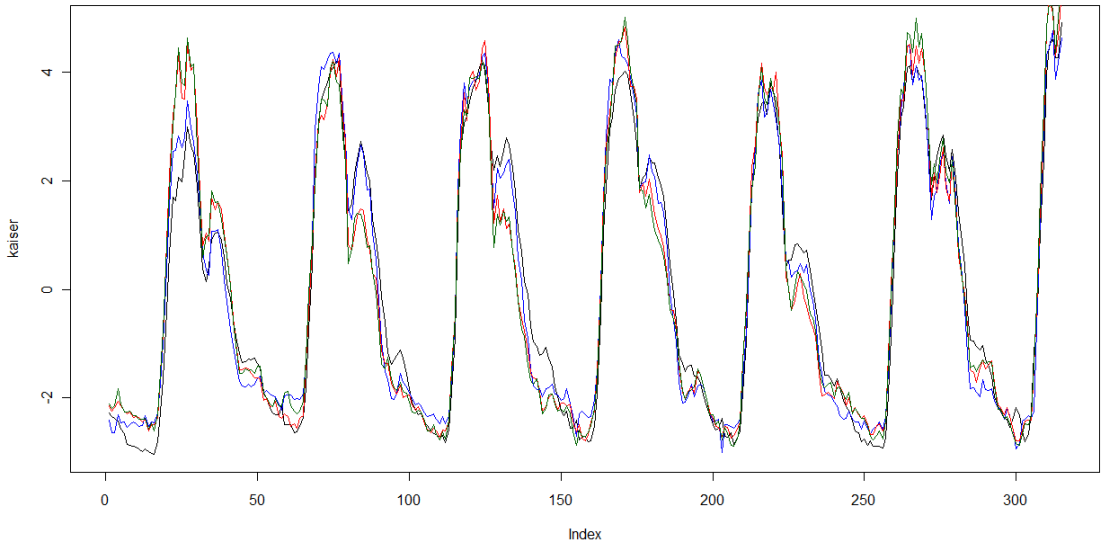


Fig. 6.3 Modelos propuestos (I)

Para observar con más claridad las diferencias entre los distintos criterios, se puede acotar la observación a un solo día (recordemos que la frecuencia de la serie es de 48 muestras/día):

```
plot(kaiser,type='l',col="black",xlim=c(0,48))
lines(var95,col="blue")
lines(var99,col="red")
lines(var100,col="darkgreen")
```

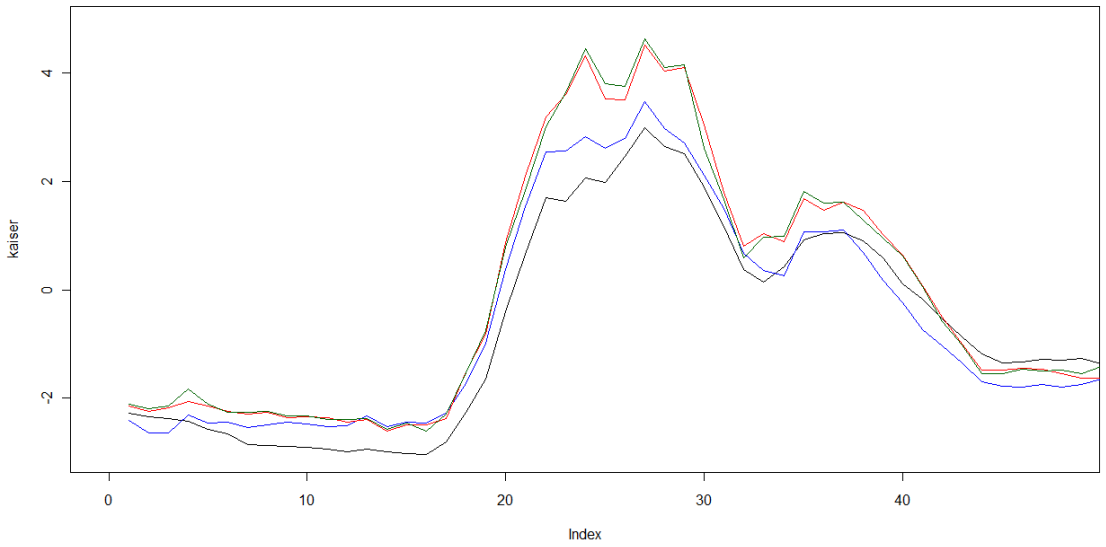


Fig. 6.4 Modelos propuestos (II)

Como se puede apreciar, a medida que los criterios de selección admiten más componentes principales, el rizado de la gráfica aumenta.

6.2.2. Predicción de la serie and_can_l

Una vez realizado este análisis, se utilizarán sus resultados para predecir el comportamiento de la serie and_can_l, que ha sido utilizada para el PCA. Como and_can_l es una serie que no presenta a priori un patrón claro reconocible (binomial, poisson, etc.), no se especificará la opción "family" de la función y se dejará a la opción por defecto, que es la gaussiana.

Teóricamente, a medida que se incrementa la cantidad de componentes principales incluidos, aumentará también la fidelidad de la predicción del modelo, a costa de un mayor coste computacional y de una simplificación menor del modelo. Así, se pueden calcular distintos modelos en función de la cantidad de componentes principales incluidos para la regresión:

```
model1 <- glm(and_can_l ~ kaiser)
model2 <- glm(and_can_l ~ var95)
model3 <- glm(and_can_l ~ var99)
model4 <- glm(and_can_l ~ var100)
```

Los datos concretos de un modelo se pueden consultar mediante la instrucción summary(). Por ejemplo, para el modelo propuesto según el criterio de Kaiser:

```
summary (model1)
```

```
Call:
glm(formula = and_can_l ~ kaiser)

Deviance Residuals:
```

```
     Min        1Q    Median        3Q       Max
-9028071  -2067055    329769   2011369  15417942

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 18634301     220125   84.65   <2e-16 ***
kaiser       5751865      93125   61.77   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 1.526333e+13)

    Null deviance: 6.3005e+16  on 314  degrees of freedom
Residual deviance: 4.7774e+15  on 313  degrees of freedom
AIC: 10460

Number of Fisher Scoring iterations: 2
```

Fig. 6.5 Datos del modelo propuesto según el criterio de Kaiser

Para ver los valores esperados para la serie and_can_l por cada modelo, sólo hay que utilizar la función *fitted* para cada uno de ellos. Para tener una idea más clara de la afinidad de las estimaciones, es mejor trazar un gráfico que superponga la serie and_can_l con sus modelos predictivos:

```
plot(and_can_l)
lines(fitted(model1), col='blue')
lines(fitted(model2), col='red')
lines(fitted(model3), col='darkgreen')
lines(fitted(model4), col='orange')
```

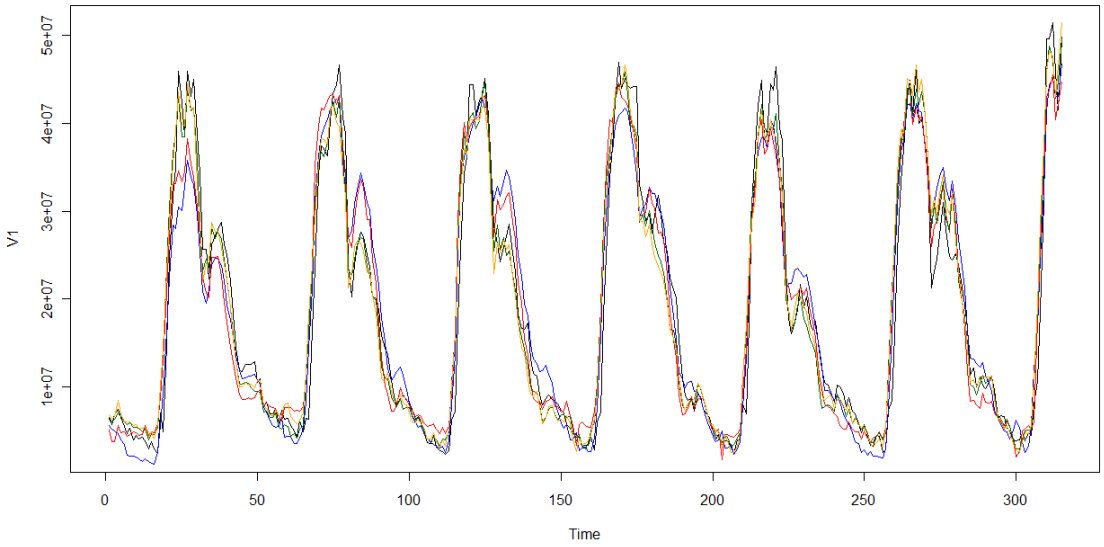


Fig. 6.6 Predicciones para la serie and_can_l (I)

Para observar con más claridad las diferencias entre las predicciones de los distintos modelos, se puede acotar la observación a un solo día:

```
plot(and_can_l,xlim=c(0,48))
lines(fitted(model1), col='blue')
lines(fitted(model2), col='red')
lines(fitted(model3), col='darkgreen')
lines(fitted(model4), col='orange')
```

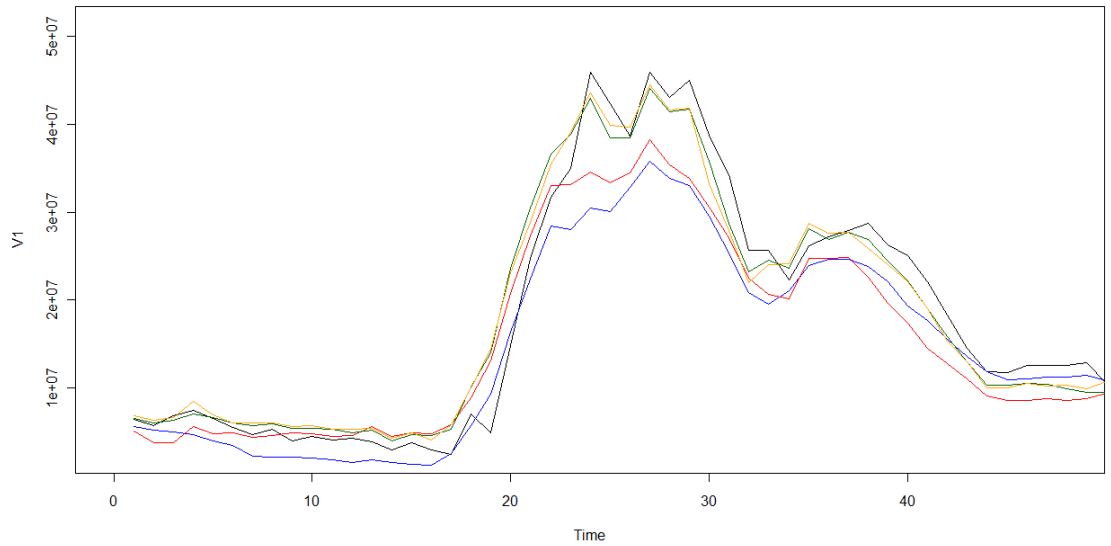


Fig. 6.7 Predicciones para la serie and_can_l (II)

En ambas gráficas se puede observar que la predicción de todos los modelos se ajusta con mucha exactitud a la forma real de la serie. Los modelos que mejor se ajustan a la realidad son los que mantienen más componentes principales, tal y como indica la teoría. En este caso particular, con pocas series temporales, el uso de muchos componentes principales no supone una carga computacional destacable. Se puede considerar que el modelo que presenta un mejor compromiso entre precisión y complejidad es el modelo 3 (99% de la varianza), que mantiene cinco de los seis componentes principales.

6.2.3. Extrapolado de la serie nac_ext

Igual que para el caso de la predicción de la serie and_can_l, se utilizarán los resultados del PCA para extrapolar el comportamiento de la serie nac_ext, que no ha sido utilizada para el análisis.

Los distintos modelos se calculan en función de la cantidad de componentes principales incluidos para la regresión:

```
model1 <- glm(nac_ext ~ kaiser)
model2 <- glm(nac_ext ~ var95)
model3 <- glm(nac_ext ~ var99)
model4 <- glm(nac_ext ~ var100)
```

A continuación se traza un gráfico que superponga la serie and_can_l con sus modelos predictivos:

```
plot(nac_ext)
```

```
lines(fitted(model1), col='blue')
lines(fitted(model2), col='red')
lines(fitted(model3), col='darkgreen')
lines(fitted(model4), col='orange')
```

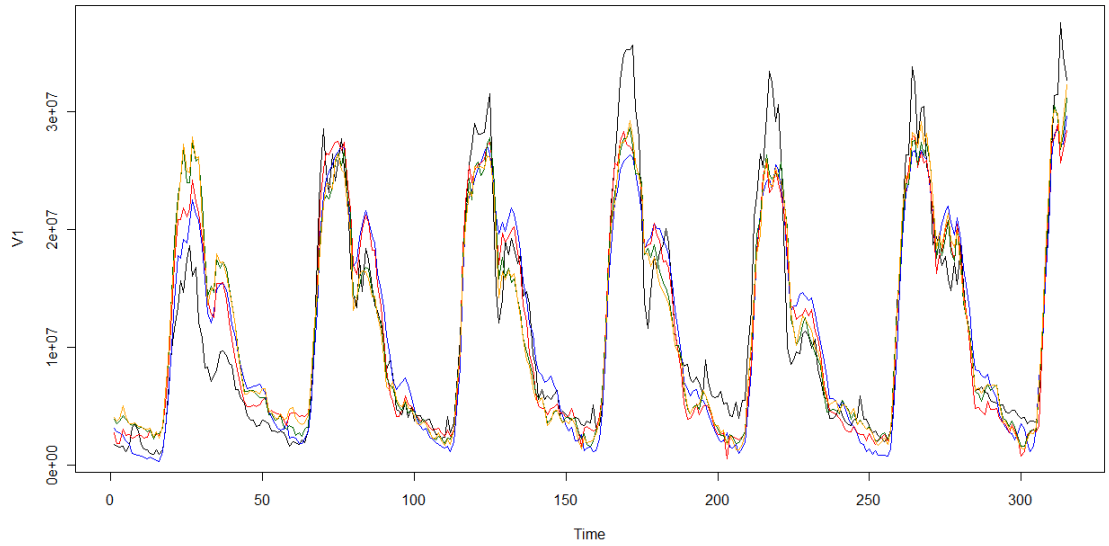


Fig. 6.8 Extrapolaciones de la serie nac_ext (I)

Para observar con más claridad las diferencias entre las predicciones de los distintos modelos, se puede acotar la observación a un solo día:

```
plot(nac_ext,xlim=c(49,96))
lines(fitted(model1), col='blue')
lines(fitted(model2), col='red')
lines(fitted(model3), col='darkgreen')
lines(fitted(model4), col='orange')
```

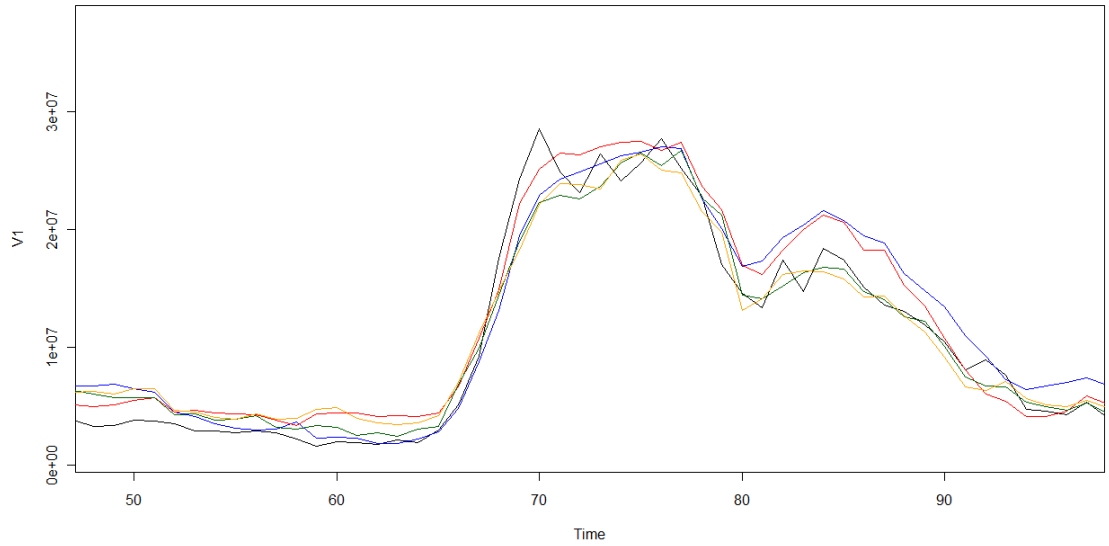


Fig. 6.9 Extrapolaciones de la serie nac_ext (II)

Debido a la gran varianza capturada por el primer componente principal, las aportaciones del resto de componentes se ven relegadas a un rizado añadido a la estructura establecida por éste, aunque como indica la teoría, estas pequeñas aportaciones ajustan mejor la forma real de la serie. Teniendo en cuenta que la serie nac_ext no se ha usado para el ACP y que el resultado no se aleja demasiado del comportamiento real, queda demostrado que se pueden extrapolar funciones mediante el modelo genérico propuesto por series con las que comparten un mismo patrón.

6.3. PCA aplicado al conjunto de enlaces de RedIRIS

En este análisis se utilizarán las 66 series correspondientes a cada dirección de cada enlace de RedIRIS. Igual que sucedía en el análisis anterior, la serie del enlace nac-ext quedará relegada a cálculos comparativos.

6.3.1. Cálculo del PCA

Una vez introducidas todas las series temporales en el entorno de R, hay que combinarlas en una misma matriz de datos:

```
fin_tot<-
cbind(s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s
19,s20,s21,s22,s23,s24,s25,s26,s27,s28,s29,s30,s31,s32,s33,s34,s35,s36
,s37,s38,s39,s40,s41,s42,s43,s44,s45,s46,s47,s48,s49,s51,s52,s53,s54,s
55,s56,s57,s58,s59,s60,s61,s62,s63,s64,s65,s66)
```

El siguiente paso es la realización del PCA. En este caso se ha escogido el método que lo calcula mediante la matriz de correlación:

```
pca <- prcomp(fin_opt, scale = TRUE)
```

El porcentaje de varianza explicado por cada componente principal es:

```
summary(pca)
```

```
Importance of components:
                          PC1    PC2   PC3   PC4    PC5    PC6    PC7
Standard deviation     5.200 2.1130 1.925 1.672 1.5595 1.4112 1.3447
Proportion of Variance 0.416 0.0687 0.057 0.043 0.0374 0.0306 0.0278
Cumulative Proportion  0.416 0.4847 0.542 0.585 0.6221 0.6528 0.6806
                           PC8    PC9   PC10   PC11   PC12   PC13
Standard deviation     1.3029 1.2425 1.1915 1.1296 1.0488 1.0222
Proportion of Variance 0.0261 0.0238 0.0218 0.0196 0.0169 0.0161
Cumulative Proportion  0.7067 0.7305 0.7523 0.7720 0.7889 0.8050
                          PC14   PC15   PC16   PC17   PC18   PC19
Standard deviation     0.9977 0.9795 0.9235 0.8839 0.8523 0.8137
Proportion of Variance 0.0153 0.0148 0.0131 0.0120 0.0112 0.0102
Cumulative Proportion  0.8203 0.8350 0.8481 0.8602 0.8713 0.8815
                          PC20    PC21    PC22    PC23    PC24    PC25
Standard deviation     0.8135 0.80459 0.76371 0.74554 0.69701 0.66876
Proportion of Variance 0.0102 0.00996 0.00897 0.00855 0.00747 0.00688
```

```
Cumulative Proportion  0.8917 0.90167 0.91064 0.91919 0.92667 0.93355
                          PC26    PC27    PC28    PC29    PC30
Standard deviation     0.64642 0.61506 0.57356 0.55001 0.53613
Proportion of Variance 0.00643 0.00582 0.00506 0.00465 0.00442
Cumulative Proportion  0.93997 0.94579 0.95086 0.95551 0.95993
                          PC31    PC32    PC33    PC34    PC35
Standard deviation     0.51488 0.50702 0.47748 0.43975 0.41239
Proportion of Variance 0.00408 0.00395 0.00351 0.00298 0.00262
Cumulative Proportion  0.96401 0.96796 0.97147 0.97445 0.97706
                          PC36    PC37   PC38   PC39    PC40    PC41
Standard deviation     0.40201 0.37214 0.3698 0.3518 0.32785 0.30737
Proportion of Variance 0.00249 0.00213 0.0021 0.0019 0.00165 0.00145
Cumulative Proportion  0.97955 0.98168 0.9838 0.9857 0.98734 0.98880
                         PC42    PC43    PC44    PC45    PC46    PC47
Standard deviation     0.2912 0.27580 0.26472 0.24664 0.23961 0.22085
Proportion of Variance 0.0013 0.00117 0.00108 0.00094 0.00088 0.00075
Cumulative Proportion  0.9901 0.99127 0.99235 0.99328 0.99417 0.99492
                         PC48    PC49    PC50    PC51    PC52    PC53
Standard deviation     0.2136 0.20180 0.18558 0.17042 0.16821 0.15819
Proportion of Variance 0.0007 0.00063 0.00053 0.00045 0.00044 0.00038
Cumulative Proportion  0.9956 0.99625 0.99678 0.99722 0.99766 0.99804
                          PC54    PC55   PC56    PC57    PC58    PC59
Standard deviation     0.15565 0.14380 0.1386 0.13343 0.11817 0.11026
Proportion of Variance 0.00037 0.00032 0.0003 0.00027 0.00021 0.00019
Cumulative Proportion  0.99842 0.99873 0.9990 0.99930 0.99952 0.99971
                          PC60    PC61  PC62    PC63    PC64    PC65
Standard deviation     0.09132 0.07682 5e-02 0.03707 0.03139 0.00729
Proportion of Variance 0.00013 0.00009 4e-05 0.00002 0.00002 0.00000
Cumulative Proportion  0.99983 0.99992 1e+00 0.99998 1.00000 1.00000
```

Fig. 6.10 Datos de la varianza para cada componente principal

A continuación, se seleccionan cuántos componentes principales se van a utilizar para generar el modelo regresivo. Según el método escogido, estos son:

- Kaiser: hay que mantener hasta el 13º componente principal, ya que es el último que cumple estrictamente que su desviación estándar es superior o igual a 1.
- 90% de la varianza: se ha de mantener hasta el 21º componente principal.
- 95% de la varianza: se ha de mantener hasta el 28º componente principal.
- 99% de la varianza: se ha de mantener hasta el 42º componente principal.
- 100% de la varianza: los 65 componentes principales.

Estas acumulaciones de componentes principales se almacenan en variables para agilizar su manejo, tal y como se ha hecho para el caso anterior.

Se pueden obtener gráficamente los resultados del análisis para cada criterio de selección. Para facilitar su comparación, los dibujaremos solapados, asignando un color a cada criterio:

```
plot(var100,type='l',col="orange")
lines(var90,col="blue")
```

```
lines(var95,col="red")
lines(var99,col="darkgreen")
lines(kaiser,col="black")
```

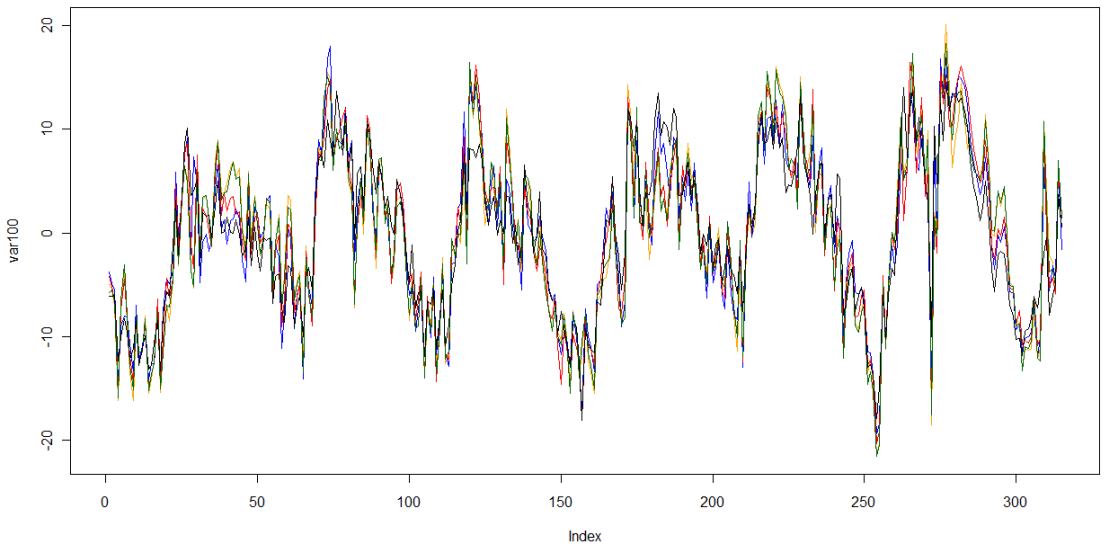


Fig. 6.11 Modelos propuestos (I)

Para observar con más claridad las diferencias entre los distintos criterios, se puede acotar la observación a un solo día:

```
plot(kaiser,type='l',col="black",xlim=c(0,48))
lines(var90,col="blue")
lines(var95,col="red")
lines(var99,col="darkgreen")
lines(var100,col="orange")
```

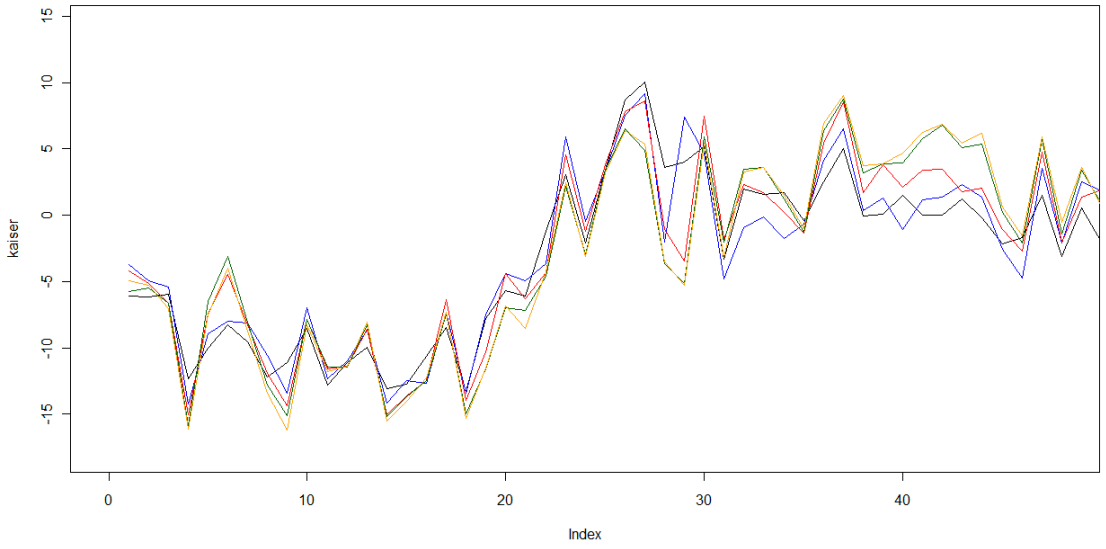


Fig. 6.12 Modelos propuestos (II)

Como se puede observar, a medida que los criterios de selección admiten más componentes principales, el rizado de la gráfica aumenta.

6.3.2. Predicción de la serie and_can_I

Los resultados del PCA se utilizarán para predecir el comportamiento de la serie s1 (and_can_I), que ha sido utilizada para el análisis.

Se pueden calcular distintos modelos en función de la cantidad de componentes principales incluidos para la regresión [27]:

```
model1 <- glm(s1 ~ kaiser)
model2 <- glm(s1 ~ var90)
model3 <- glm(s1 ~ var95)
model4 <- glm(s1 ~ var99)
model5 <- glm(s1 ~ var100)
```

Para ver los valores esperados para la serie and_can_I por cada modelo, solo hay que utilizar la función *fitted* para cada uno de ellos. Para tener una idea más clara de las estimaciones, es mejor trazar un gráfico que superponga la serie and_can_I con todos sus modelos predictivos:

```
plot(s1)
lines(fitted(model5), col='turquoise')
lines(fitted(model2), col='red')
lines(fitted(model3), col='darkgreen')
lines(fitted(model4), col='orange')
lines(fitted(model1), col='blue')
```

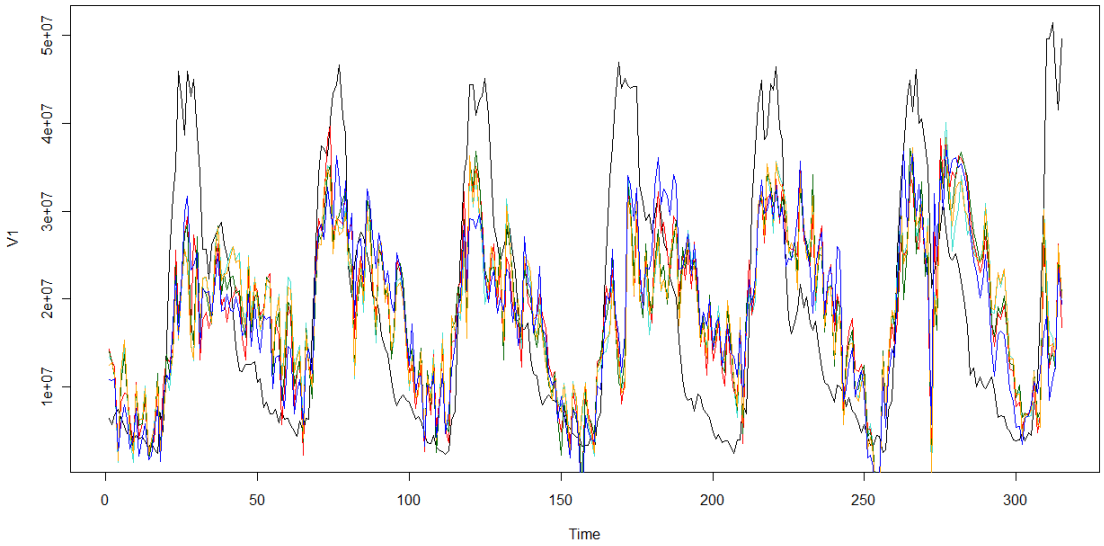


Fig. 6.13 Predicciones para la serie and_can_I (I)

Para observar con más claridad las diferencias entre las predicciones de los distintos modelos, se puede acotar la observación a un solo día:

```
plot(s1,xlim=c(49,96))
```

```
lines(fitted(model1), col='blue')
lines(fitted(model2), col='red')
lines(fitted(model3), col='darkgreen')
lines(fitted(model4), col='orange')
lines(fitted(model5), col='turquoise')
```

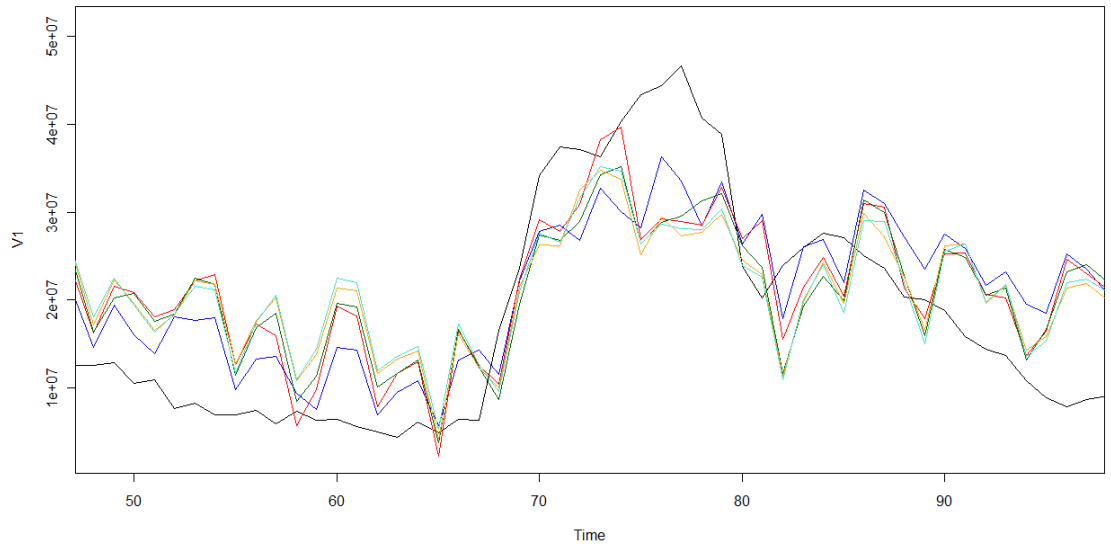


Fig. 6.14 Predicciones para la serie and_can_I (II)

Si comparamos estos resultados con los obtenidos para el escenario reducido de seis series, se puede observar que las predicciones han perdido parte de su calidad. No obstante, hemos de tener en cuenta que gran parte de las series incluidas en el PCA son de back-up y su patrón es muy errático o inexistente. Por ejemplo, la serie s28 (enlace cat-val) tiene el aspecto:

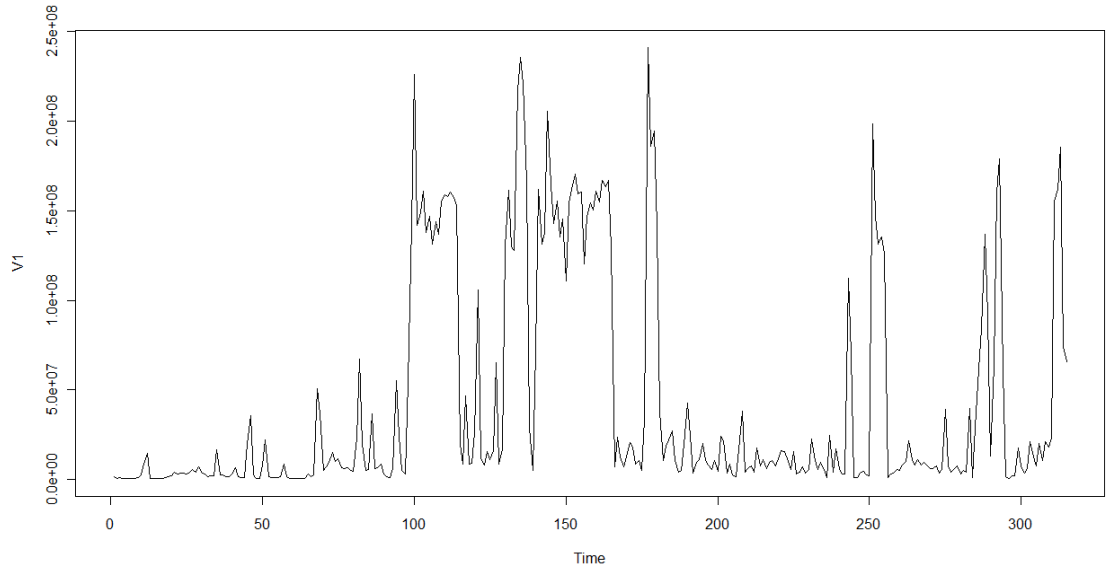


Fig. 6.15 Serie cat_val

Este hecho convierte la predicción realizada en válida ya que, marginalmente, los modelos propuestos se pueden seguir utilizando para predecir, aunque de manera aproximada, el comportamiento general de la serie. De hecho, los

modelos ponen de manifiesto la existencia de una estacionalidad diaria compartida por todas las series temporales, así como la tendencia a una mayor actividad matutina y el progresivo decremento de utilización de los enlaces a lo largo del lapso temporal de un día.

6.3.3. Extrapolado de la serie nac_ext

Igual que para el caso de la predicción de la serie and_can_l, se utilizarán los resultados del PCA para extrapolar el comportamiento de la serie s50 (nac_ext), que no ha sido utilizada para el análisis.

Los distintos modelos se calculan en función de la cantidad de componentes principales incluidos para la regresión:

```
model1 <- glm(s50 ~ kaiser)
model2 <- glm(s50 ~ var90)
model3 <- glm(s50 ~ var95)
model4 <- glm(s50 ~ var99)
model5 <- glm(s50 ~ var100)
```

A continuación se traza un gráfico que superponga la serie and_can_l con sus modelos predictivos:

```
plot(s50)
lines(fitted(model5), col='turquoise')
lines(fitted(model2), col='red')
lines(fitted(model3), col='darkgreen')
lines(fitted(model4), col='orange')
lines(fitted(model1), col='blue')
```

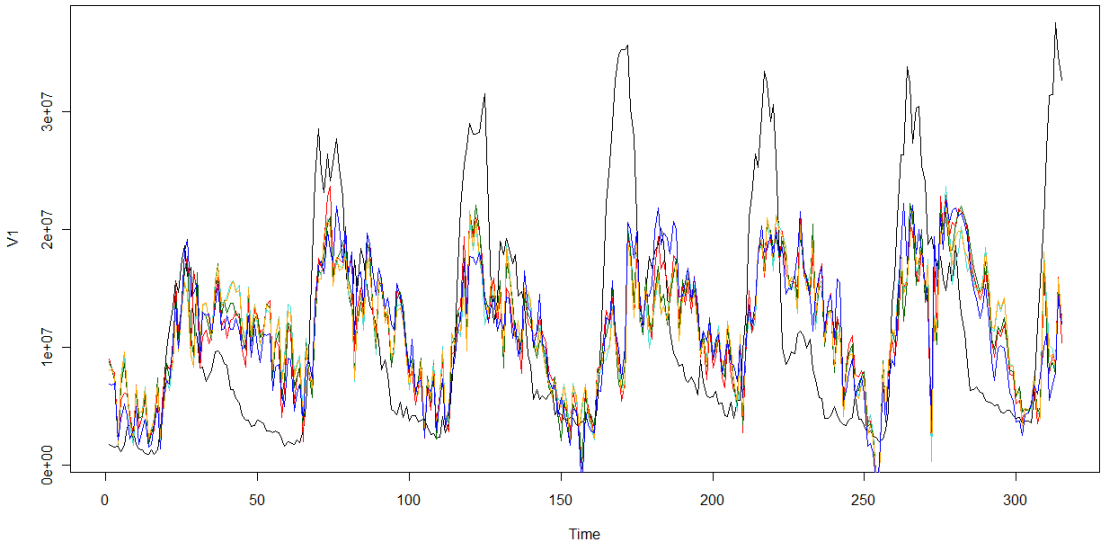


Fig. 6.16 Extrapolaciones para la serie nac_ext (I)

Para observar con más claridad las diferencias entre las predicciones de los distintos modelos, se puede acotar la observación a un solo día:

```
plot(s50,xlim=c(49,96))
```

```
lines(fitted(model1), col='blue')
lines(fitted(model2), col='red')
lines(fitted(model3), col='darkgreen')
lines(fitted(model4), col='orange')
lines(fitted(model5), col='turquoise')
```

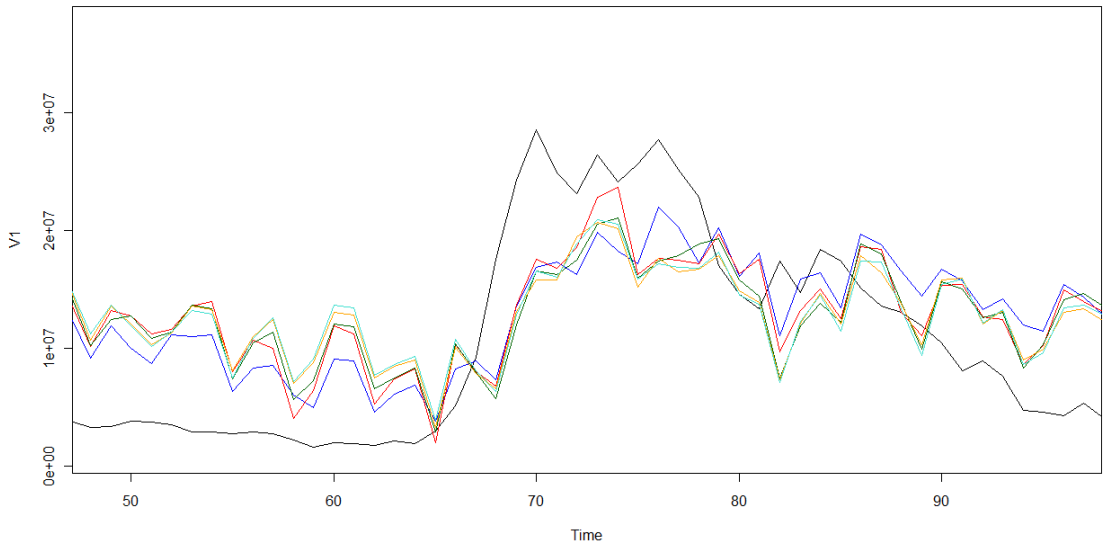


Fig. 6.17 Extrapolaciones para la serie nac_ext (II)

Aunque los modelos no pueden capturar con precisión la magnitud del flujo de bytes del enlace, sí que proporcionan una visión aproximada del comportamiento de la serie. De hecho, la estacionalidad y la tendencia que presenta la serie real están debidamente representadas en las extrapolaciones.

Estos resultados no se diferencian mucho de los obtenidos para la predicción de la serie and_can_l, lo que sugiere que la utilización del PCA sigue siendo un sistema válido para extrapolar series en escenarios extensos como el presentado.

CAPÍTULO 7. CONCLUSIONES Y LÍNEAS FUTURAS

7.1. Conclusiones

Este trabajo se ha centrado en la aplicación de las técnicas de pronóstico estadístico a la predicción del comportamiento del tráfico de una red IP. Recordemos que los dos objetivos del presente trabajo eran, por una parte, la aplicación de las técnicas aprendidas al caso real de una red operativa (RedIRIS), con la finalidad de comprobar la calidad de las predicciones que proporcionan y, por otra parte, la valoración de la utilidad de las técnicas estudiadas para la planificación del crecimiento de una red.

Este documento sienta las bases tanto teóricas (los fundamentos teóricos del TSA y del PCA) como prácticas (mediante la aplicación al entorno de R) sobre las que realizar futuros trabajos relacionados con la predicción de tráfico en redes IP.

A lo largo de este documento hemos podido comprobar la eficacia de estas técnicas para predecir los comportamientos futuros de series referentes a flujos de datos reales, así como para extrapolar enlaces completos ajenos al análisis utilizado. Los resultados no engañan: los modelos predictivos han conseguido proporcionar unos pronósticos que a veces sorprenden por su exactitud.

Aunque los cálculos realizados se han limitado a un ámbito de corto y medio plazo porque facilitaban la tarea de comprobación (ya que no hacían necesario esperar largos períodos de tiempo para obtener confirmaciones a nuestras sospechas), las técnicas son perfectamente escalables y la única limitación se encuentra en la capacidad de computación de las máquinas utilizadas, ya que cuanto más complejos sean los cálculos más tiempo requieren. Por ejemplo, para llevar a cabo este trabajo se ha utilizado un PC con procesador Intel® Core™ 2 Duo processor T5250 a 1.5 GHz y 2GB de memoria RAM y el tiempo medio de computación para un TSA de no más de 4 órdenes constituye un tiempo de computación de 4 segundos, mientras que el de un PCA de 66 componentes es prácticamente instantáneo.

Finalmente, queremos agradecer a RedIRIS la aportación de los datos utilizados a lo largo de este trabajo, ya que nos han brindado la oportunidad de medir la utilidad de las técnicas estudiadas sobre un caso real.

7.2. Impacto medioambiental

La buena utilización de los recursos que proporciona una red puede reducir notablemente el gasto energético que comporta su utilización. Si se utilizan estrategias de predicción combinadas con un adecuado balanceo de carga, se podría programar la desconexión de los equipos que no sean estrictamente necesarios para el buen funcionamiento de la red. En este sentido se han

desarrollado iniciativas como el desarrollo de los routers virtuales. Por otra parte, también se podría evitar el gasto energético provocado por las congestiones de red o la caída de un nodo mediante la predicción de situaciones críticas.

7.3. Líneas futuras de desarrollo

A pesar de que se han alcanzado los objetivos iniciales del proyecto, existe un gran número de posibles líneas de futuro desarrollo orientadas a la aplicación de las técnicas estudiadas:

- El uso de las técnicas predictivas dedicadas a series temporales que alcancen varios años vista puede proporcionar una buena base sobre la que realizar cálculos precisos del crecimiento particular de una red como RedIRIS o GÉANT. Sobre estos datos se pueden analizar tendencias y ajustar el presupuesto dedicado a infraestructuras de manera más efectiva a las necesidades reales.
- En el campo de la gestión de redes, se puede crear software que gestione internamente las técnicas de predicción y sea capaz de proponer soluciones automáticas ante la aparición de situaciones críticas, como por ejemplo realizar balanceo de carga ante la saturación de un enlace. Un par de ejemplos de software de este tipo son el simulador TOTEM [23], que permite emular, desde un punto de vista teórico, el comportamiento de la red y el simulador perfSONAR [24], la herramienta de monitorización de la red académica europea GÉANT [2], que podría dar estimaciones en tiempo real de la carga prevista de los enlaces y disparar automáticamente algoritmos de optimización de rutas. En un TFC realizado recientemente en la EETAC se dan indicaciones de cómo se podría modificar perfSONAR [25] para cumplir éste propósito.
- En el campo de la seguridad de redes, se puede desarrollar software de monitorización de red basado en las técnicas de predicción ya que por ejemplo, el uso de modelos predictivos puede ayudar en la detección de ataques de denegación de servicio (basados en sobrecargar de tráfico ciertos servidores para que caigan), ya que causan patrones de comportamiento extraños.

BIBLIOGRAFÍA

[1] RedIRIS, http://www.rediris.es/rediris/index.html.es

[2] GÉANT, Network http://www.geant.net/

[3] ESPANIX, http://www.espanix.net/

[4] CATNIX, http://www.catnix.net/

[5] RRDtool, Home Site http://oss.oetiker.ch/rrdtool/

[6] RRDtool, Wikipedia http://en.wikipedia.org/wiki/RRDtool

[7] Round Robin, Wikipedia http://en.wikipedia.org/wiki/Round-robin

[8] MRTG, Wikipedia http://en.wikipedia.org/wiki/Multi_Router_Traffic_Grapher

[9] Weathermap, http://www.rediris.es/conectividad/weathermap/

[10] Cryer, J., Chang, K., *Time Series Analysis with applications in R*, Springer, (2008, 2nd ed.).

[11] Box-Jenkins method, Wikipedia http://en.wikipedia.org/wiki/Box-Jenkins

[12] MLE, Wikipedia http://en.wikipedia.org/wiki/Maximum_likelihood_estimator

[13] AIC, Wikipedia http://en.wikipedia.org/wiki/Akaike_information_criterion

[14] R Project, http://cran.r-project.org/

[15] SNR, Wikipedia http://en.wikipedia.org/wiki/Signal-to-noise_ratio

[16] PCA, Wikipedia http://en.wikipedia.org/wiki/Principal_component_analysis

[17] Análisis en Componentes Principales, http://www.oldemarrodriguez.com/yahoo_site_admin/assets/docs/cap2.143111051.pdf

[18] PCA, http://www.cesma.usb.ve/~lbravo/co6111/clase1.pdf

[19] Balasch, I., “Anàlisi Espai-Temporal Multiresolució de Matrius de Teletràsit”. Treball Fi de Carrera EPSC (2010).

[20] Torres, J., “Modelling of traffic matrices with multiresolution analysis techniques”. Treball Fi de Carrera EPSC (2009).

[21] Principal Components Analysis (PCA) R tutorial, http://www.uga.edu/strata/software/pdf/pcaTutorial.pdf

[22] Rincón, D.,Balasch, I., **"**Análisis espacio-temporal multiresolución de matrices de tráfico". IX Jornadas de Ingenieria Telemática Jitel 2010 (Valladolid), Septiembre 2010.

[23] TOTEM Project, http://totem.run.montefiore.ulg.ac.be/

[24] perfSONAR, http://www.perfsonar.net/

[25] Sáez, V., "Desplegament, anàlisi i millora d'un sistema distribuït de mesures en xarxes de nova generació". Treball Fi de Carrera EPSC (2010).

[26] Minguillón, I., "Medida y análisis de matrices de tráfico en redes IP" Treball Fi de Carrera EPSC (2010).

[27] PCA logistic regression, http://yatani.jp/HCIstats/PCA

GLOSARIO

ACF	Autocorrelation Function
AIC	Akaike Information Criterion
AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
DOS	Denial of Service attack
EACF	Extended Autocorrelation Function
HTML	Hyper Text Markup Language
IP	Internet Protocol
MA	Moving Average
MLE	Maximum Likelihood Estimation
MRTG	Multi Router Traffic Grapher
MSE	Mean Square Error
PACF	Partial Autocorrelation Function
PCA	Principal Component Analysis
PCR	Principal Component Regression
RRA	Round Robin Archive
RRD	Round Robin Database
SARIMA	Seasonal Autoregressive Integrated Moving Average
SNMP	Simple Network Management Protocol
SVD	Singular Value Decomposition
TSA	Time Series Analysis
XML	Extensible Markup Language

ANEXOS

A.I. El lenguaje de programación R

En este punto se detalla la descripción del lenguaje de programación R así como las funciones utilizadas en los cálculos del trabajo.

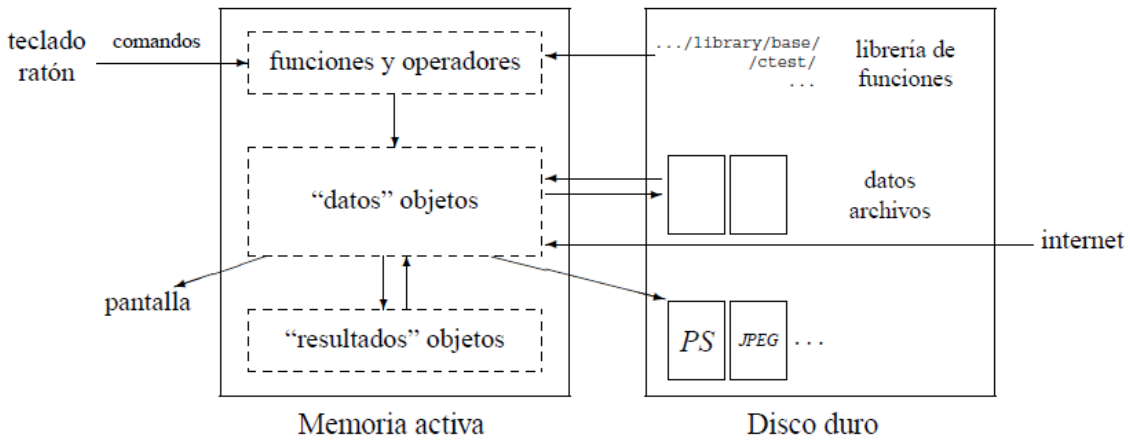
A.I.1. Introducción a R

R es un sistema para análisis estadísticos y gráficos creado en 1993 por Ross Ihaka y Robert Gentleman del Departamento de Estadística de la Universidad de Auckland. Tiene una naturaleza doble de programa y lenguaje de programación y se considera un dialecto del lenguaje S creado por los Laboratorios AT&T Bell ya que de hecho, la mayoría del código escrito para S puede funcionar en R sin ser alterado.

R se distribuye gratuitamente bajo los términos de la *GNU General Public Licence* y su desarrollo y distribución es responsabilidad del *R Development Core Team*. Está disponible en código fuente escrito principalmente en C y algunas rutinas en Fortran (esencialmente para máquinas Unix y Linux) y como archivos binarios precompilados (para Windows, Debian, Mandrake, RedHat, SuSe, Macintosh y Alpha Unix). Los archivos necesarios para instalar R, ya sea desde las fuentes o binarios pre-compilados, se distribuyen desde el sitio de internet *Comprehensive R Archive Network* (CRAN) junto con las instrucciones de instalación. En este entorno se pueden encontrar también los *package* (librerías) que amplían las funcionalidades de R. Hay que tener en cuenta que algunas librerías modifican el comportamiento de funciones pertenecientes a librerías ya instaladas.

R posee muchas funciones para análisis estadísticos y gráficos. Estos últimos pueden ser visualizados de manera inmediata en su propia ventana y se pueden guardar en varios formatos como jpg, png, bmp, ps, pdf, emf, pictex y xfig, siempre teniendo en cuenta que los formatos disponibles dependen del sistema operativo utilizado.

Los resultados de análisis estadísticos se muestran en la pantalla, y algunos resultados intermedios (como valores *P*-, coeficientes de regresión, residuales,…) se pueden guardar, exportar a un archivo, o ser utilizados en análisis posteriores. El lenguaje R permite al usuario, por ejemplo, programar bucles (o loops en inglés) para analizar conjuntos sucesivos de datos. También es posible combinar en un solo programa diferentes funciones estadísticas para realizar análisis más complejos. El esquema general del funcionamiento de R es:



A.I.2. Funciones para el TSA

La librería "TSA" proporciona las funciones y los conjuntos de datos necesarios para realizar los cálculos detallados en el libro "Time Series Analysis with Applications in R (second edition)" de Jonathan Cryer y Kung-Sik Chan. Las funciones proporcionadas en ésta librería consisten tanto en funciones desarrolladas específicamente para el análisis de series temporales como en extensiones de las funciones incluidas en la librería "stats", que suele formar parte de las librerías incluidas en todas las versiones de R.

- **La función acf():** calcula (y por defecto muestra gráficamente) las estimaciones de la función de autocorrelación o de autocovariancia. La versión incluida en la librería "TSA" es una extensión de la incluida en la librería "stats". Su sintaxis es la siguiente:

```
acf(x, lag.max = NULL, type = c("correlation", "covariance",
"partial"), plot = TRUE, na.action = na.fail, demean = TRUE,
drop.lag.0 = TRUE, ...)
```

`x`	Un objeto de clase serie temporal (univariante o multivariante), vector o matriz numéricos.
`lag.max`	El número máximo de retrasos sobre los que calcular la función de autocorrelación. Su valor por defecto es 10*log10(N/m), donde N es el número de observaciones y m es el número de series.
`type`	Un string de caracteres que indican qué tipo de función se va a realizar. Los valores permitidos son "correlation" (el valor por defecto), "covariance" y "partial" (sólo utilizado por la función pacf()).
`plot`	Un valor lógico que indica si la función será representada gráficamente. Su valor por defecto es TRUE.
`na.action`	La función que se llama para tratar valores perdidos. Se puede utilizar la función na.pass.
`demean`	Un valor lógico que sirve para indicar si se deben utilizar la media de las muestras. Su valor por defecto es TRUE.

`drop.lag.0` Un valor lógico que permite indicar si el retraso 0 debe ser ignorado. Su valor por defecto es TRUE.

`...` Argumentos para pasar al método plot().

La función retorna un objeto de tipo “acf” que contiene los siguientes atributos:

`lag` Un array de tres dimensiones que contiene los retrasos para los que se estima la función acf().

`acf` Un array con tantas dimensiones como retrasos que contiene las autocorrelaciones/autocovarianzas estimadas.

`type` El tipo de correlación (igual que el argumento type).

`n.used` El número de observaciones de las series temporales.

`series` El nombre de la serie del argumento x.

`snames` Los nombres de las series para el caso de series temporales multivariadas.

Una vez retornado el objeto, se pueden aplicar métodos específicos para mostrar el contenido concreto de los atributos del objeto o ver estos atributos de manera gráfica:

- El método summary(): muestra las características de los atributos que constituyen el objeto devuelto por acf(). Su sintaxis es:
  `summary(object)`
  `object` El objeto de clase “acf” devuelto por la función.
- Los accesos a los atributos: permiten acceder de manera manual a los atributos del objeto retornado por acf(). El funcionamiento del sistema de acceso a los atributos es general y su sintaxis es `nombre_objeto$atributo`. Donde “nombre_objeto” es el nombre con el que se ha almacenado el objeto retornado por la función acf() y donde “atributo” es el nombre del atributo cuyo contenido se quiere mostrar.
- La función plot(): sirve para representar los resultados de la función acf(). Es la misma función que representa gráficamente los resultados de la función si el argumento plot está puesto a TRUE. En el eje y se indica el valor de las autocorrelaciones/autocovarianzas calculadas para cada retraso, mientras que en el eje x se enumeran los retrasos.

- **La función pacf():** calcula (y por defecto muestra gráficamente) las estimaciones de la función de autocorrelación parcial. Esta función es una extensión de la función acf() ya que de hecho, el cálculo lo realiza la función acf() con el argumento type fijado a “partial”. La asignación de este nombre permite simplificar su sintaxis y facilita su reconocimiento. Su sintaxis es la siguiente:

```
pacf(x, lag.max = NULL, plot = TRUE, na.action = na.fail, ...)
```

`x`	Un objeto de clase serie temporal (univariante o multivariante), vector o matriz numéricos.
`lag.max`	El número máximo de retrasos sobre los que calcular la función de autocorrelación parcial. Su valor por defecto es 10*log10(N/m), donde N es el número de observaciones y m es el número de series.
`plot`	Un valor lógico que indica si la función será representada gráficamente. Su valor por defecto es TRUE.
`na.action`	La función que se llama para tratar valores perdidos. Se puede utilizar la función na.pass.
`...`	Argumentos para pasar al método plot().

Esta función también retorna un objeto de clase “acf” que contiene los mismos atributos que se han definido anteriormente:

`lag`	Un array de tres dimensiones que contiene los retrasos para los que se estima la función acf().
`acf`	Un array con tantas dimensiones como retrasos que contiene las autocorrelaciones parciales estimadas.
`type`	El tipo de correlación (igual que el argumento type). Fijado a “partial”.
`n.used`	El número de observaciones de las series temporales.
`series`	El nombre de la serie del argumento x.
`snames`	Los nombres de las series para el caso de series temporales multivariadas.

Una vez retornado el objeto, se pueden aplicar métodos específicos para mostrar el contenido concreto de los atributos del objeto o ver estos atributos de manera gráfica:

- El método summary(): muestra las características de los atributos que constituyen el objeto devuelto por pacf(). Su sintaxis es:

  ```
  summary(object)
  ```

  `object` El objeto de clase “acf” devuelto por la función.
- Los accesos a los atributos: permiten acceder de manera manual a los atributos del objeto retornado por pacf(). El funcionamiento del sistema de acceso a los atributos es general y su sintaxis es nombre_objeto$atributo. Donde “nombre_objeto” es el nombre con el que se ha almacenado el objeto retornado por la función pacf() y donde “atributo” es el nombre del atributo cuyo contenido se quiere mostrar.
- La función plot(): sirve para representar los resultados de la función pacf(). Es la misma función que representa gráficamente los resultados de la función si el argumento plot está puesto a TRUE. En el eje y se indica el valor de las autocorrelaciones parciales calculadas para cada retraso, mientras que en el eje x se enumeran los retrasos.

- **La función eacf():** calcula y muestra gráficamente los resultados de la función de autocorrelación muestral extendida. Esta función pertenece a la librería "TSA" y su sintaxis es la siguiente:

```
eacf(z, ar.max = 7, ma.max = 13)
```

`z`	Un objeto de clase serie temporal.
`ar.max`	El orden máximo AR. Su valor por defecto es 7.
`ma.max`	El orden máximo MA. Su valor por defecto es 13.

La función retorna un objeto de tipo "eacf" que contiene los siguientes atributos:

`eacf`	Una matriz con los resultados de la función de autocorrelación muestral extendida.
`ar.max`	El orden máximo AR (igual que el argumento ar.max).
`ma.max`	El orden máximo MA (igual que el argumento ma.max).
`symbol`	La matriz de símbolos que indica la significancia de cada valor de la eacf.

La función eacf() muestra por pantalla, de manera automática, una tabla con los datos de la matriz symbol, que contiene los valores eacf codificados para facilitar su interpretación. Los valores considerados significativos se denotan con una x y los considerados no significativos con un 0.

Una vez retornado el objeto, se pueden aplicar métodos específicos para mostrar el contenido concreto de los atributos del objeto:

- El método summary(): muestra las características de los atributos que constituyen el objeto devuelto por eacf(). Su sintaxis es:

```
summary(object)
```

`object` El objeto de clase "eacf" devuelto por la función.

- Los accesos a los atributos: permiten acceder de manera manual a los atributos del objeto retornado por eacf(). El funcionamiento del sistema de acceso a los atributos es general y su sintaxis es `nombre_objeto$atributo`. Donde "nombre_objeto" es el nombre con el que se ha almacenado el objeto retornado por la función eacf() y donde "atributo" es el nombre del atributo cuyo contenido se quiere mostrar.

- **La función arima():** ajusta un modelo ARIMA a una serie temporal univariante. Esta función es idéntica a la función arimax (también de la librería "TSA"), que extiende la capacidad de la función arima de la librería "stats" permitiendo la incorporación de funciones de transferencia y de outliers (valores atípicos) innovativos y additivos. Nótese que en la computación del AIC, el número de parámetros excluye la varianza del

ruido. El nombre de arima se ha mantenido por compatibilidad, pero la sintaxis es la siguiente:

```
arima(x, order = c(0, 0, 0), seasonal = list(order = c(0, 0,
0), period = NA), xreg = NULL, include.mean = TRUE,
transform.pars = TRUE, fixed = NULL, init = NULL, method =
c("CSS-ML", "ML", "CSS"), n.cond, optim.control = list(), kappa
= 1e+06, io = NULL, xtransf, transfer = NULL)
```

`x`	La serie temporal univariante.
`order`	Los órdenes de la parte regular del modelo ARIMA: los tres componentes (p, d, q) son el orden AR, el grado de diferenciación y el orden MA.
`seasonal`	Los órdenes de la parte estacional del modelo ARIMA más el período (que si no se especifica se deja por defecto al valor de la frecuencia). El vector se llena por orden estándar: P, D y Q.
`xreg`	Un vector o matriz de regresores externos opcional (covariables) que debe tener el mismo número de columnas que x.
`include.mean`	Un valor lógico que puesto a cierto, incorpora un término de intercepción en el modelo (sólo aplicable en los modelos estacionarios). Su valor por defecto es TRUE.
`transform.pars`	Un valor lógico que puesto a cierto, transforma los parámetros AR para asegurar la estacionariedad. No se utiliza para el método “CSS”. Su valor por defecto es TRUE.
`fixed`	Un vector numérico opcional que indica qué coeficientes son fijos o libres.
`init`	Un vector opcional de valores iniciales para los parámetros.
`method`	El método de estimación: máxima verosimilitud (ML) o suma condicional de cuadrados (CSS). Los valores permitidos son “CSS-ML”, “ML” y “CSS”, en función del método que se quiera especificar. Por defecto (y a menos que falten valores) se usa la suma condicional de cuadrados para encontrar los valores iniciales y a continuación máxima verosimilitud (CSS-ML).
`n.cond`	Sólo se utiliza si se aplica suma condicional de cuadrados. Indica el número de observaciones iniciales que se han de ignorar. Se ignora de todos modos si es menos que el máximo retraso de un término AR.
`optim.control`	Parámetros de control para el proceso de optimización.
`kappa`	La varianza previa de las observaciones anteriores en un modelo diferenciado.
`io`	Una lista de puntos de tiempo en los cuales el modelo puede presentar un valor atípico innovativo. El punto de tiempo del valor atípico se puede dar como punto de tiempo absoluto o como c(a,b), es decir, bº “mes” del aº

"año" donde cada año tiene una frecuencia (x) meses, suponiendo que x es una serie temporal.

`xtransf` Es una matriz en la que cada columna contiene una covariable que afecta a la respuesta de la serie temporal en términos de un filtro ARMA de orden (p,q). Es decir que si Z es una covariable tal que su efecto en la serie de tiempo es $(theta_0 + theta_1 B + ... + theta_{q-1} B^{q-1})/(1-phi_1 B - ... - phi_p B^p) Z_t$ en particular, si p=0 y q=1, esto especifica una relación simple de regresión, lo que debe ser incluido en xreg y no aquí. Nótese que el filtro empieza con cero como valores iniciales.

`transfer` Una lista que consiste en los órdenes ARMA para cada transferencia (de retrasos distribuidos) covariable.

La función retorna un objeto de clase "arima" que contiene los siguientes atributos:

`coef` Un vector de los coeficientes de regresión AR y MA que pueden ser extraídos mediante el método coef().

`sigma2` El MLE de la varianza de las innovaciones.

`var.coef` La matriz de varianza estimada de los coeficientes coef que han sido extraídos por el método vcov().

`loglik` El logaritmo de la verosimilitud (de los datos diferenciados), o una aproximación de éste.

`arma` Una forma compacta de especificación, en forma de vector que proporciona los números de los coeficientes de AR, MA, AR estacional, MA estacional, el período y el número de diferenciación regular y estacional.

`aic` El valor AIC correspondiente al logaritmo de la verosimilitud. Sólo válido para los ajustes del método "ML".

`residuals` Las innovaciones ajustadas.

`call` La definición específica de la función utilizada.

`series` El nombre de la serie x.

`code` El valor de convergencia retornado por optim().

`n.cond` El número de observaciones iniciales no usadas para el ajuste.

`model` Una lista representando el Filtro Kalman usado en el ajuste.

Una vez retornado el objeto, se pueden aplicar métodos específicos para mostrar el contenido concreto de los atributos del objeto:

- El método summary(): muestra las características de los atributos que constituyen el objeto devuelto por arima(). Su sintaxis es:

  ```
  summary(object)
  ```

  `object` El objeto de clase "arima" devuelto por la función.

- Los accesos a los atributos: permiten acceder de manera manual a los atributos del objeto retornado por arima(). El funcionamiento

del sistema de acceso a los atributos es general y su sintaxis es nombre_objeto$atributo. Donde "nombre_objeto" es el nombre con el que se ha almacenado el objeto retornado por la función arima() y donde "atributo" es el nombre del atributo cuyo contenido se quiere mostrar.

- **La función plot.arima():** muestra gráficamente los datos de la serie temporal y sus predicciones con límites de predicción del 95%. La versión incluida en la librería "TSA" es una extensión de la incluida en la librería "stats". Su sintaxis es la siguiente:

```
plot(x, n.ahead = 12, col = "black", ylab = object$series, lty
= 2, n1, newxreg, transform, Plot=TRUE, ...)
```

`x`	Un modelo ARIMA ajustado.
`n.ahead`	El número de pasos avanzados predichos. Por defecto es de 12.
`col`	El color de los límites de predicción.
`ylab`	Etiqueta del eje y.
`n1`	El instante de tiempo que conforma el punto de partida de la gráfica. Por defecto se asigna al punto temporal más antiguo.
`newxreg`	Una matriz de covarianzas sobre el período de predicción.
`transform`	Función utilizada para transformar los pronósticos y sus límites de predicción. Si falta, no se lleva a cabo ninguna transformación. Esta opción es útil si el modelo fue ajustado a los datos transformados y se desea obtener los pronósticos para la escala original. Por ejemplo, si el modelo fue ajustado con un logaritmo de los datos, entonces transform = exp trazaría el gráfico de los pronósticos y sus límites de predicción en la escala original.
`Plot`	La creación del gráfico será suprimido si Plot se pone a FALSE. El valor por defecto es TRUE.
`...`	Argumentos para pasar al método plot().

- **La función rstandard.Arima():** computa los residuales estandarizados para un modelo ajustado ARIMA. Esta función pertenece a la librería "TSA" y su sintaxis es la siguiente:

```
rstandard(model, ...)
```

`model`	El modelo ajustado por la función ARIMA.
`...`	No usado. Se mantiene por consistencia con el modelo genérico.

La función retorna una serie temporal compuesta por los residuales estandarizados. Como serie temporal, se le pueden aplicar los métodos específicos para mostrar el contenido concreto de los atributos del objeto o ver estos atributos de manera gráfica:

  - El método summary(): muestra las características de la serie

temporal de los residuales. Su sintaxis es:

```
summary(object)
```

`object` El objeto de clase “eacf” devuelto por la función.

- Los accesos a los atributos: permiten acceder de manera manual a los atributos de la serie retornada por rstandard(). El funcionamiento del sistema de acceso a los atributos es general y su sintaxis es `nombre_objeto$atributo`. Donde “nombre_objeto” es el nombre con el que se ha almacenado el objeto retornado por la función rstandard() y donde “atributo” es el nombre del atributo cuyo contenido se quiere mostrar.

A.I.3. Funciones para el PCA

R proporciona tres funciones para llevar a cabo un análisis de componentes principales o ACP (PCA en inglés): prcomp() y princomp() de la librería “stats” y pca() de la librería “labdsv”. Aunque las tres funciones pueden realizar un análisis basado en la matriz de covarianzas o en la de correlaciones y todas computan los mismos valores, el método que utilizan es diferente y mientras que prcomp() y pca() realizan una descomposición de valores singulares, princomp() lleva a cabo un análisis de vectores y valores propios.

- **La función prcomp():** se caracteriza por ser una rutina numéricamente estable y se considera la función preferida para llevar a cabo los análisis de componentes principales debido a su exactitud numérica. En caso de utilizar la matriz de covarianzas, estas son computadas con el divisor N-1 habitual. Su sintaxis es la siguiente:

```
prcomp(x, retx = TRUE, center = TRUE, scale. = FALSE,tol= NULL)
```

`x` La matriz numérica o compleja que provee los datos para realizar el ACP. Las muestras se disponen en filas y los atributos en columnas.

`retx` Un valor lógico que indica si las variables rotadas deben ser retornadas.

`center` Un valor lógico que indica si las variables deben ser modificadas para que estén centradas a cero. Alternativamente, se puede proporcionar un vector de longitud igual al número de columnas de x. Éste valor se pasa a scale.

`scale.` Un valor lógico que indica si las variables deben ser escaladas para tener una varianza unitaria antes de que se realice el análisis. Por defecto su valor es FALSE por consistencia con S, pero en general se recomienda el escalado. Alternativamente, se puede proporcionar un vector de longitud igual al número de columnas de x. Éste valor se pasa a la función scale(). Para realizar un ACP basado en la matriz de correlaciones, el valor ha de ser verdadero, mientras que para utilizar la matriz de covarianzas el valor ha de ser falso.

`tol` Un valor que sirve para indicar la magnitud a partir de la cual los componentes deben ser omitidos. Se omitirán los componentes

cuyas desviaciones estándar sean menores o iguales a tol veces la desviación estándar del primer componente. Con la disposición por defecto a nulo, no se omite ningún componente. Otras configuraciones para tol pueden ser tol = 0 o tol = sqrt(.Machine$double.eps), que omitirían los componentes constantes.

La función retorna un objeto de tipo “prcomp” que contiene los siguientes atributos:

`sdev`	Las desviaciones estándar de los componentes principales (las raíces cuadradas de los valores propios de la matriz de covarianzas/correlaciones, aunque el cálculo se realice con los valores singulares de la matriz de datos).
`rotation`	La matriz de cargas de la variable (una matriz cuyas columnas contienen los vectores propios). Los signos de esta matriz son arbitrarios y pueden diferir entre diferentes programas para ACP o incluso para diferentes compilaciones de R.
`x`	Son los datos proyectados en la nueva base de vectores propios.
`center, scale`	El centrado y el escalado utilizado.

Una vez retornado el objeto, se pueden aplicar métodos específicos para mostrar el resultado del análisis, el contenido concreto de los atributos del objeto, ver estos atributos de manera gráfica o incluso utilizar el análisis realizado para calcular las proyecciones de matrices de datos sobre la base del espacio reducido:

- El método summary(): muestra los componentes principales por orden de importancia mediante la desviación estándar, la proporción de la varianza y la proporción acumulativa. Su sintaxis es:

  ```
  summary(object, dim = length(object$sdev))
  ```

`object`	El objeto de clase “prcomp” extraído del análisis.
`dim`	El número de dimensiones a devolver.

- Los accesos a los atributos: permiten acceder de manera manual a los atributos del objeto retornado por prcomp(). El funcionamiento del sistema de acceso a los atributos es general y su sintaxis es `nombre_objeto$atributo`. Donde “nombre_objeto” es el nombre con el que se ha almacenado el objeto retornado por la función prcomp() y donde “atributo” es el nombre del atributo cuyo contenido se quiere mostrar.
- El método predict(): permite proyectar una matriz de datos sobre la nueva base de vectores propios encontrada durante el análisis. Su sintaxis es:

  ```
  predict(object, newdata)
  ```

`object`	El objeto de clase “prcomp” extraído del análisis.
`newdata`	Es una matriz de datos opcional. En caso de ser

incluida, se calculará la proyección de esos datos sobre la base encontrada durante el análisis. Cabe señalar que ésta matriz ha de mantener el formato de los datos de la matriz "x" utilizada para el análisis de la función prcomp(). Si no se incluye, el cálculo se lleva a cabo con los datos de la matriz "x".

- La función plot(): muestra gráficamente los datos del método "summary". En el eje y se indica el valor de las varianzas para cada componente principal, mientras que en el eje x se enumeran los componentes principales por orden de importancia.
- La función biplot(): muestra el plano formado por los dos componentes principales más significativos. Contiene dos tipos de datos:
  - Un conjunto de puntos que representan las proyecciones de las observaciones sobre el plano de las dos componentes principales estandarizadas y con varianza unitaria.
  - Un conjunto de vectores que representan las coordenadas de las variables originales (loadings) en los dos primeros componentes estandarizados.

  En un gráfico biplot se cumple lo siguiente:
  - Las distancias euclídeas entre los puntos del plano biplot equivalen, aproximadamente, a las distancias de Mahalanobis entre las observaciones originales.
  - Los ángulos entre los vectores representan aproximadamente la correlación entre las variables.

- **La función princomp():** es ligeramente menos estable aunque presenta más atributos. El cálculo se realiza mediante la función eigen() sobre la matriz de covarianza o de correlación determinada mediante cov() o cor() respectivamente. Esto se realiza por compatibilidad con el resultado proporcionado por el lenguaje S-PLUS. En caso de utilizar la matriz de covarianzas, estas son computadas con el divisor N. Su sintaxis es la siguiente:

```
princomp(x, cor = FALSE, scores = TRUE, covmat = NULL,subset =
rep(TRUE, nrow(as.matrix(x))))
```

`x` La matriz numérica o compleja que provee los datos para realizar el ACP. Las muestras se disponen en filas y los atributos en columnas.

`cor` Un valor lógico que indica si el cálculo se ha de realizar usando la matriz de correlación o la matriz de covarianza.

`scores` Un valor lógico que indica si se ha de calcular el valor de cada componente principal.

`covmat` Una matriz de covarianza o lista de covarianza que de ser aportada se usa en lugar de la matriz de covarianza de x.

`subset` Un vector opcional usado para seleccionar filas (observaciones) de la matriz de datos x.

La función retorna un objeto de tipo “princomp” que contiene los siguientes atributos:

`sdev` Las desviaciones estándar de los componentes principales (las raíces cuadradas de los valores propios de la matriz de covarianzas/correlaciones).

`loadings` La matriz de cargas de la variable (una matriz cuyas columnas contienen los vectores propios). Los signos de esta matriz son arbitrarios y pueden diferir entre diferentes programas para ACP o incluso para diferentes compilaciones de R.

`center` Las medias que son sustraídas.

`scale` El escalado aplicado a cada variable.

`n.obs` El numero de observaciones.

`scores` Son los datos proyectados en la nueva base de vectores propios.

Al objeto devuelto por la función, se le pueden aplicar métodos específicos para mostrar el resultado del análisis, el contenido concreto de los atributos del objeto, ver estos atributos de manera gráfica o incluso utilizar el análisis realizado para calcular las proyecciones de matrices de datos sobre los resultados:

- El método summary(): muestra los componentes principales por orden de importancia mediante la desviación estándar, la proporción de la varianza y la proporción acumulativa. Su sintaxis es:

  ```
  summary(object, dim = length(object$sdev))
  ```

  `object` El objeto de clase “princomp” extraído del análisis.

  `dim` El número de dimensiones a devolver.

- Los accesos a los atributos: permiten acceder de manera manual a los atributos del objeto retornado por princomp(). El funcionamiento del sistema de acceso a los atributos es general y su sintaxis es `nombre_objeto$atributo`. Donde “nombre_objeto” es el nombre con el que se ha almacenado el objeto retornado por la función prcomp() y donde “atributo” es el nombre del atributo cuyo contenido se quiere mostrar.
- El método loadings(): proporciona los datos del atributo “loadings”. El resultado es el mismo que el obtenido mediante un acceso manual.
- El método predict(): permite proyectar una matriz de datos sobre la nueva base de vectores propios encontrada durante el análisis. Su sintaxis es:

  ```
  predict(object, newdata)
  ```

  `object` El objeto de clase “princomp” extraído del análisis.

  `newdata` Es una matriz de datos opcional. En caso de ser incluida, se calculará la proyección de esos datos sobre la base encontrada durante el análisis. Cabe señalar que ésta matriz ha de mantener el formato de

los datos de la matriz “x” utilizada para el análisis de la función princomp(). Si no se incluye, el cálculo se lleva a cabo con los datos de la matriz “x”.

- La función plot(): muestra gráficamente los datos del método “summary”. En el eje y se indica el valor de las varianzas para cada componente principal, mientras que en el eje x se enumeran los componentes principales por orden de importancia.
- La función biplot(): muestra el plano formado por los dos componentes principales más significativos. Contiene dos tipos de datos:
  - Un conjunto de puntos que representan las proyecciones de las observaciones sobre el plano de las dos componentes principales estandarizadas y con varianza unitaria.
  - Un conjunto de vectores que representan las coordenadas de las variables originales (loadings) en los dos primeros componentes estandarizados.

  En un gráfico biplot se cumple lo siguiente:
  - Las distancias euclídeas entre los puntos del plano biplot equivalen, aproximadamente, a las distancias de Mahalanobis entre las observaciones originales.
  - Los ángulos entre los vectores representan aproximadamente la correlación entre las variables.

- **La función pca():** está basada en la función prcomp() (para hacer su uso consistente con otras funciones de la librería) y por lo tanto es opuesta a la función princomp(). Aunque permite limitar a un número fijo las dimensiones calculadas, pca() no puede automatizar éste proceso mediante umbrales como hace prcomp(). No obstante, mantiene las etiquetas más convencionales “scores” y “loadings” en lugar de “x” y “rotation”. Su sintaxis es la siguiente:

```
pca(mat, cor = FALSE, dim = min(nrow(mat),ncol(mat)))
```

`mat` La matriz con los datos a los que se les va a aplicar el ACP. Las muestras se disponen en filas y los atributos en columnas.

`cor` Un valor lógico que indica si el cálculo se ha de realizar usando la matriz de correlación (valor a TRUE) o la matriz de covarianza (valor a FALSE).

`dim` El número de dimensiones a devolver.

La función retorna un objeto de tipo “pca” que contiene los siguientes atributos:

`scores` Una matriz de coordenadas de las muestras en el espacio reducido.

`loadings` Una matriz de las contribuciones de las variables a los ejes del espacio reducido (una matriz cuyas columnas contienen los vectores propios).

`sdev` Un vector de las desviaciones estándar para cada dimensión (las raíces cuadradas de los valores propios de la matriz de covarianzas/correlaciones).

Al objeto devuelto, se le pueden aplicar métodos específicos para mostrar el resultado del análisis, el contenido concreto de los atributos del objeto o ver estos atributos de manera gráfica:

- El método summary(): muestra los componentes principales por orden de importancia mediante la desviación estándar, la proporción de la varianza y la proporción acumulativa. Su sintaxis es:

  ```
  summary(object, dim = length(object$sdev))
  ```

  `object` El objeto de clase “pca” extraído del análisis.

  `dim` El número de dimensiones a devolver.

- Los accesos a los atributos: permiten acceder de manera manual a los atributos del objeto retornado por pca(). El funcionamiento del sistema de acceso a los atributos es general y su sintaxis es `nombre_objeto$atributo`. Donde “nombre_objeto” es el nombre con el que se ha almacenado el objeto retornado por la función pca() y donde “atributo” es el nombre del atributo cuyo contenido se quiere mostrar.
- El método scores(): proporciona los datos del atributo “scores”. Su sintaxis es:

  ```
  scores(x, labels = NULL, dim = length(x$sdev))
  ```

  `x` El objeto de clase “pca” extraído del análisis.

  `labels` Un vector de etiquetas opcional para identificar puntos.

  `dim` El número de dimensiones a devolver.

  Si no se especifican limitadores, el resultado es el mismo que el obtenido mediante un acceso manual.
- El método loadings(): proporciona los datos del atributo “loadings”. Su sintaxis es:

  ```
  loadings(x, dim = length(x$sdev), digits = 3,
  cutoff = 0.1)
  ```

  `x` El objeto de clase “pca” extraído del análisis.

  `dim` El número de dimensiones a devolver.

  `digits` El número de dígitos con los que se escriben los resultados.

  `cutoff` Umbral establecido para eliminar valores pequeños.

  Si no se especifican limitadores, el resultado es el mismo que el obtenido mediante un acceso manual.
- La función plot(): muestra el plano formado por los dos componentes principales más significativos con todos los puntos proyectados sobre este plano.
- La función varplot.pca(): muestra gráficamente los datos del método “summary”. En el eje y se indica el valor de las varianzas

para cada componente principal, mientras que en el eje x se enumeran los componentes principales por orden de importancia. Si volvemos a pulsar enter, aparece un segundo gráfico que muestra las varianzas acumulativas para cada componente principal. Esta gráfica está normalizada a uno, que representa el total de las varianzas.

- **La función glm():** se utiliza para ajustar modelos lineales generalizados, especificados por una descripción simbólica de la predicción lineal y una descripción de la distribución del error. Pertenece a la librería “stats” y su sintaxis es la siguiente:

```
glm(formula, family = gaussian, data, weights, subset,
na.action, start = NULL, etastart, mustart, offset, control =
list(...), model = TRUE, method = "glm.fit", x = FALSE, y =
TRUE)
```

`formula`	Un objeto de clase “formula” (o uno que pueda ser aplicado a la clase) que proporciona una descripción simbólica del modelo que ha de ser ajustado.
`family`	Una descripción del error de distribución y la función de enlace que se utilizarán en el modelo. Se puede indicar en forma de string y admite las funciones de distribución habituales: “gaussian” (activada por defecto), “binomial”, “poisson”, etc.
`data`	Una lista de datos opcional que contiene las variables del modelo. Si no se especifica, las variables se obtienen del entorno en el que se llamó a glm().
`weights`	Un vector opcional de los “pesos previos” para ser usados en el proceso de ajuste. Debe ser NULL o un vector numérico.
`subset`	Un vector opcional que especifica un subconjunto de observaciones que han de ser utilizadas en el proceso de ajuste.
`na.action`	Una función que indica que debe ocurrir en cuando los datos contienen NAs (números no disponibles). Por defecto se marca a na.fail (que indique fallo), aunque también se puede marcar a na.omit, na.exclude y NULL.
`start`	Los valores iniciales para los parámetros del predictor lineal.
`etastart`	Los valores iniciales para el predictor lineal.
`mustart`	Los valores iniciales para el vector de medias.
`offset`	Sirve para especificar de manera opcional un componente conocido a priori que ha de ser incluido en el predictor lineal durante el ajuste. Debe ser NULL o un vector numérico de la misma longitud que el número de casos. Si se especifica más de un offset, se utiliza su suma.
`control`	Una lista de parámetros para controlar el proceso de ajuste.
`model`	Un valor lógico que indica si el frame del modelo debe ser incluido como un componente del valor devuelto.
`method`	El método que ha de ser usado en el ajuste del modelo. El método por defecto es glm.fit() y utiliza el sistema de mínimos

cuadrados iterativamente reponderados (IWLS).

`x, y` Valores lógicos que indican si el vector de respuesta y la matriz del modelo usados en el proceso de ajuste deben ser retornados como componentes del valor retornado.

La función retorna un objeto de clase “glm” y contiene los siguientes atributos:

`coefficients` Un vector de coeficientes.

`residuals` Los residuales utilizados, que son los residuales de la iteración final del ajuste IWLS.

`fitted.values` Los valores medios ajustados, obtenidos mediante la transformación lineal de los predictores mediante la inversa de la función de enlace.

`rank` El rango numérico del modelo lineal ajustado.

`family` La familia del objeto utilizado.

`linear.predictors` El ajuste lineal en la escala de enlace.

`deviance` Hasta una constante, menos el doble del logaritmo de la verosimilitud máximo. Cuando es sensible, la constante se elige de modo que un modelo saturado tenga desviación cero.

`aic` Una versión del Criterio de Información de Akaike.

`null.deviance` La desviación para el modelo nulo, comparable con “deviance”.

`iter` El número de iteraciones usadas en el IWLS.

`weights` Los pesos utilizados, que son los pesos utilizados en la iteración final del ajuste IWLS.

`prior.weights` Los pesos iniciales.

`df.residual` Los grados de libertad residuales

`df.null` Los grados de libertad residuales para el modelo nulo.

`y` Si se solicita (la opción por defecto), el vector “y” utilizado.

`x` Si se solicita, la matriz modelo.

`model` Si se solicita (la opción por defecto), el frame del modelo.

`converged` Indica si el algoritmo IWLS ha convergido.

`boundary` Indica si el valor ajustado está en el límite de los valores alcanzables.

`call` La definición exacta de la función utilizada.

`formula` La “formula” proporcionada.

`terms` Los términos del objeto usado.

`data` Los datos del argumento “data”.

`offset` El vector de offset utilizado.

`control` El valor del argumento “control” utilizado.

`method`	El nombre de la función de filtrado utilizada, normalmente glm.fit().
`contrasts`	Los contrastes utilizados (donde sea relevante)
`xlevels`	Un registro de los niveles de los factores utilizados en el ajuste (donde sea relevante)
`na.action`	La información retornada por model.frame para el trato especial de los NAs (donde sea relevante).

Una vez retornado el objeto, se pueden aplicar métodos específicos para mostrar el resultado del análisis, el contenido concreto de los atributos del objeto, ver estos atributos de manera gráfica o incluso utilizar el análisis realizado para calcular las proyecciones de matrices de datos sobre la base del espacio reducido:

- El método summary(): muestra la información más significativa del modelo ajustado. Su sintaxis es:
  `summary(object)`
  `object` El objeto de clase “glm” obtenido previamente.
- Los accesos a los atributos: permiten acceder de manera manual a los atributos del objeto retornado por glm(). El funcionamiento del sistema de acceso a los atributos es general y su sintaxis es `nombre_objeto$atributo`. Donde “nombre_objeto” es el nombre con el que se ha almacenado el objeto retornado por la función glm() y donde “atributo” es el nombre del atributo cuyo contenido se quiere mostrar.
- La función plot(): muestra cuatro gráficos y se suceden clicando sobre el gráfico. En orden de aparición, tenemos:
  - Diagrama Residuales vs. Valores Predictivos: muestra la media de residuales asociada a cada valor que aparece en la serie temporal.
  - Diagrama QQ entre la desviación estándar de los residuales y las cantidades teóricas: muestra el ajuste entre las desviaciones estándar registradas y las desviaciones esperadas para una distribución normal teórica. La recta teórica representa la trayectoria que presentaría un ajuste exacto entre la desviación estándar registrada y las cantidades teóricas (representativa de una distribución normal exacta).
  - Diagrama escala-localización: muestra la comparativa entre la raíz de las desviaciones estándar contra la escala de valores de la serie predicha.
  - Diagrama Residuales vs. Apalancamiento (Leverage): muestra el valor de los residuales enfrentados al cálculo del apalancamiento. En una línea discontinua aparece representada la distancia de Cook.

- **La función fitted():** también conocida como fitted.values, es una función que extrae los valores ajustados de objetos retornados por funciones de modelado. Pertenece a la librería “stats” y su sintaxis es la siguiente:

```
fitted(object) o también fitted.values(object)
```

`object` El objeto del que se van a extraer los valores ajustados.

La función muestra por pantalla un listado de los valores extraídos del objeto. No obstante, estos valores se suelen concatenar con más funciones como plot() y así mostrarlos por pantalla.

A.II. Instalación de nfdump y nfsen

Este manual de instalación está basado en gran parte en el referenciado en el TFC de Iván Minguillón [26]. En esta versión, está actualizado a las necesidades de Open SuSe 12.

flow-tools

Para compilar ft2nfdump, aplicación perteneciente a nfdump que lee, convierte y guarda los datos procedentes de NetFlow, se necesita el código fuente de la versión 0.68 de las flow-tools ya que la cabecera necesaria para ello no está en la 0.68.4.3; es por esto que se han de obtener las dos versiones de los siguientes enlaces:
http://www.splintered.net/sw/flow-tools/ (v0.68)
http://code.google.com/p/flow-tools/downloads/detail?name=flow-tools-0.68.4.3.tar.bz2&can=2&q= (v0.68.4.3)

Se descomprimen ambos ficheros
```
ivan@debian:~$ tar -zxvf flow-tools-0.68.tar.gz
ivan@debian:~$ tar -jxvf flow-tools-0.68.4.3.tar.bz2
```

Cambiar al directorio donde se encuentra el código fuente de la versión 0.68.4.3
```
ivan@debian:~$ cd flow-tools-0.68.4.3/
```

Configurar, compilar e instalar las flow-tools
```
ivan@debian:~/flow-tools-0.68.4.3$ ./configure
ivan@debian:~/flow-tools-0.68.4.3$ make
ivan@debian:~/flow-tools-0.68.4.3$ su
Password:
debian:/home/ivan/flow-tools-0.68.4.3# make install
```

Una vez instaladas, hay que copiar el fichero libft.a al directorio flow-tools-0.68.4.3/lib
```
ivan@debian:~$ cp /usr/local/flow-tools/lib/libft.a flow-tools-0.68.4.3/lib/
```

También es necesario copiar el fichero ftbuild.h del código fuente de flow-tools-0.68/src a flow-tools-0.68.4.3/src, y así poder compilar posteriormente nfdump incluyendo la aplicación ft2nfdump
```
ivan@debian:~$ cp flow-tools-0.68/src/ftbuild.h flow-tools-0.68.4.3/src/
```

rrdtool

- Cambio para OpenSuSE 11.2: Utilizar Yast para descargar rrdtools y el módulo de perl MailTools

Para la instalación de rrdtool se necesitan una serie de paquetes que se instalan automáticamente como dependencias gracias al programa aptitude, front-end del Advanced Packaging Tool (APT) de Debian:
```
debian:~# aptitude install libpango1.0-dev libxml2-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
The following NEW packages will be installed:
  debhelper{a} html2text{a} libcairo2-dev{a} libdirectfb-dev{a} libdirectfb-
  extra{a} libexpat1-dev{a} libfontconfig1-dev{a} libfreetype6-dev{a}
  libglib2.0-dev{a} libice-dev{a} libjpeg62-dev{a} libmpeg3-1{a} libmpeg3-
  dev{a} libpango1.0-dev libpixman-1-dev{a} libpng12-dev{a} libpthread-
  stubs0{a} libpthread-stubs0-dev{a} libsm-dev{a} libsysfs-dev{a} libx11
```

```
  dev{a} libxau-dev{a} libxcb-render-util0-dev{a} libxcb-render0-dev{a}
  libxcb1-dev{a} libxdmcp-dev{a} libxext-dev{a} libxft-dev{a} libxml2-dev
  libxrender-dev{a} x11proto-core-dev{a} x11proto-input-dev{a} x11proto-kb
  dev{a} x11proto-render-dev{a} x11proto-xext-dev{a} xtrans-dev{a}
```

El código fuente de rrdtool está disponible en http://oss.oetiker.ch/rrdtool/pub/?M=D y se necesita la versión 1.4.2 (rrdtool-1.4.2.tar.gz)

Se descomprime el fichero y se cambia al nuevo directorio para configurar rrdtool
```
ivan@debian:~$ tar -zxvf rrdtool-1.4.2.tar.gz
ivan@debian:~$ cd rrdtool-1.4.2/
```

Se compilará el programa en /home/ivan/rrdtool-1.4.2 (donde está el código fuente) y se instalará en /usr/local/rrdtool, directorio que es necesario crear:
```
ivan@debian:~/rrdtool-1.4.2$ su
Password:
debian:/home/ivan/rrdtool-1.4.2# mkdir /usr/local/rrdtool
```

Una vez se tiene la estructura correcta para la compilación e instalación de rrdtool, se han de guardar las dos variables de entorno:
```
ivan@debian:~/rrdtool-1.4.2$ BUILD_DIR=/home/ivan/rrdtool-1.4.2/
ivan@debian:~/rrdtool-1.4.2$ INSTALL_DIR=/usr/local/rrdtool/
```

Configurar, compilar e instalar
```
ivan@debian:~/rrdtool-1.4.2$ ./configure --prefix=$INSTALL_DIR
ivan@debian:~/rrdtool-1.4.2$ make
ivan@debian:~/rrdtool-1.4.2$ su
Password:
debian:/home/ivan/rrdtool-1.4.2# make install
```

nfdump

Para la instalación de nfdump son necesarios los siguientes paquetes con sus respectivas dependencias: flex, byacc y libghc6-zlib-dev
```
debian:~# aptitude install flex byacc libghc6-zlib-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
The following NEW packages will be installed:
  byacc flex ghc6{a} haskell-utils{a} libghc6-zlib-dev libgmp3-dev{a}
  libgmpxx4ldbl{a} libreadline5-dev{a}
```

El código fuente de nfdump se encuentra disponible en http://sourceforge.net/projects/nfdump/
La versión compatible con las flow-tools es la 1.6 (nfdump-1.6.tar.gz)

Se descomprime el fichero y se cambia al nuevo directorio para configurar nfdump
```
ivan@debian:~$ tar -zxvf nfdump-1.6.tar.gz
ivan@debian:~$ cd nfdump-1.6/
```

Al configurar nfdump con la opción nfprofile, necesaria para instalar nfsen, configure espera encontrar rrdtool en /usr con rrd.h en /usr/include y librrd en /usr/lib
El directorio donde está rrdtool se puede especificar con la opción `--enable-nfprofile --with-rrdpath=[/usr]`, pero tanto la cabecera rrd.h como las librerías hay que copiarlas a los directorios correspondientes
```
debian:~# cp /usr/local/rrdtool/include/rrd.h /usr/include/
debian:~# cp /usr/local/rrdtool/lib/librrd* /usr/lib/
```

Configurar nfdump incluyendo el convertidor ft2nfdump y el nfprofile necesario para nfsen:
```
ivan@debian:~/nfdump-1.6$ ./configure --enable-nfprofile --with-rrdpath=/usr/local/rrdtool/lib/ --enable-ftconv --with-ftpath=/home/ivan/flow-tools-0.68.4.3/
```

Compilar e instalar
```
ivan@debian:~/nfdump-1.6$ make
ivan@debian:~/nfdump-1.6$ su
Password:
debian:/home/ivan/nfdump-1.6# make install
```

nfsen

Para la instalación de nfsen son necesarios los siguientes paquetes con sus respectivas dependencias: php5 php-net-socket libapache2-mod-php5 libapache2-mod-perl2 librrds-perl
```
debian:~# aptitude install php5 php-net-socket libapache2-mod-php5 libapache2-mod-perl2 librrds-perl
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
The following NEW packages will be installed:
  apache2{a} apache2-mpm-prefork{a} apache2-utils{a} apache2.2-bin{a}
  apache2.2-common{a} libapache2-mod-perl2 libapache2-mod-php5
  libapache2-reload-perl{a} libapr1{a} libaprutil1{a}
  libaprutil1-dbd-sqlite3{a} libaprutil1-ldap{a} libbsd-resource-perl{a}
  libdb4.8{a} libdevel-symdump-perl{a} libperl5.10{a} librrds-perl php-net-socket php-pear{a} php5 php5-cli{a} php5-common{a} php5-suhosin{a}
```

El código fuente de nfsen se encuentra disponible en http://sourceforge.net/projects/nfsen/
La versión compatible con nfdump es la 1.3.2 (nfsen-1.3.2.tar.gz)

Se descomprime el fichero y se cambia al nuevo directorio
```
ivan@debian:~$ tar -zxvf nfsen-1.3.2.tar.gz
ivan@debian:~$ cd nfsen-1.3.2/
```

Hacer una copia del fichero de configuración nfsen-dist.conf que hay dentro de etc renombrándolo a nfsen.conf
```
ivan@debian:~/nfsen-1.3.2$ cp etc/nfsen-dist.conf etc/nfsen.conf
```

Editar nfsen.conf de acuerdo a las necesidades del sistema
```
ivan@debian:~/nfsen-1.3.2$ vi etc/nfsen.conf
```

Modificar $BASEDIR para que apunte a una partición con suficiente espacio, ya que es donde se almacenarán los datos procedentes de NetFlow
```
# Required for default layout
$BASEDIR = "/data1/nfsen";
```

Hay que crear una carpeta con el nombre de la ruta BASEDIR
/data1/nfsen (en este caso)

Definir con qué usuario se ejecutará nfcapd y los procesos NetFlow
```
# BASEDIR unrelated vars:
#
# Run nfcapd as this user
# This may be a different or the same uid than your web server.
# Note: This user must be in group $WWWGROUP, otherwise nfcapd
#       is not able to write data files!
$USER    = "netflow";

# user and group of the web server process
# All netflow processing will be done with this user
```

```
$WWWUSER  = "www-data";
$WWWGROUP = "www-data";
```

- Cambio para OpenSuSE 11.2: modificar la línea HTMLDIR por $HTMLDIR = "/srv/www/htdocs/nfsen/"; (el fichero por defecto donde se encuentra la página index.html del servidor apache)

Al tener instalado el servidor Apache, tanto el usuario como el grupo www-data ya existen en el sistema, por lo que sólo es necesario crear el usuario netflow y añadirlo al grupo www-data

```
debian:~# useradd -g www-data netflow
```

Para comprobar que el usuario pertenece al grupo

```
debian:~# groups netflow
netflow : www-data
```

- Cambio para OpenSuSE 11.2: crear manualmente el grupo "www-data" antes de crear al usuario "netflow"

Ejecutar como root el script de instalación pasándole como opción el fichero de configuración

```
debian:/home/ivan/nfsen-1.3.2# ./install.pl etc/nfsen.conf
Check for required Perl modules: All modules found.
Upgrade from version '1.3.2' installed at Mon Feb 22 17:37:53 2010
Setup NfSen:
Version: 1.3.2: $Id: install.pl 24 2007-11-21 09:12:03Z phaag $
```

Indicar la ruta donde se encuentra perl, por defecto /usr/bin/perl

```
Perl to use: [/usr/bin/perl]
```

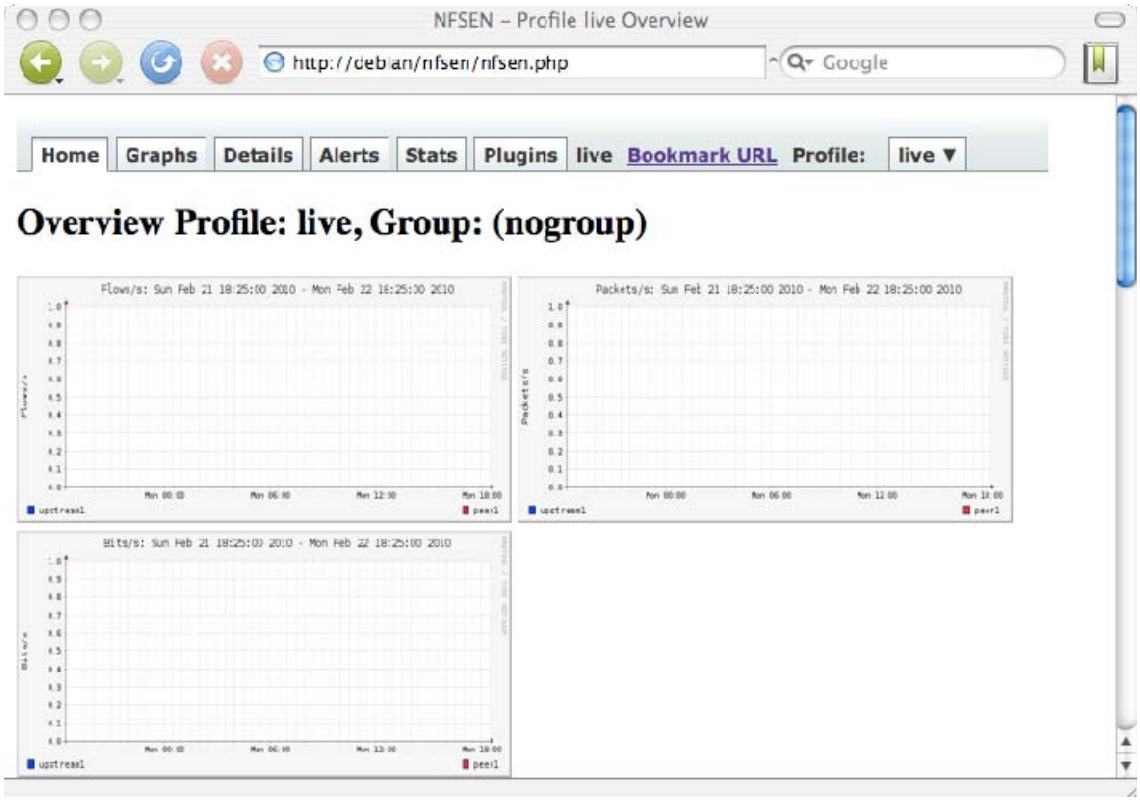
Una vez que se ha instalado correctamente, enlazar $BASEDIR/bin/nfsen a /etc/init.d/nfsen

```
debian:~# ln -s /data1/nfsen/bin/nfsen /etc/init.d/nfsen
```

Iniciar nfsen

```
debian:~# /etc/init.d/nfsen start
Starting nfcapd: upstream1[2445] peer1[2448].
Starting nfsend.
debian:~#
```

Abrir en un navegador la dirección http://nombreservidor/nfsen/nfsen.php
(http://localhost/nfsen/nfsen.php o http://linux-axem/nfsen/nfsen.php p.ej)



Si no se tenía instalado con anterioridad Apache con soporte para php, es recomendable reiniciar el servicio y recargar los módulos de php. La forma más sencilla es reiniciando la máquina.

Posible error de nfsen:

Si en el fichero nfsen.conf se ha especificado otro usuario o grupo para ejecutar nfcapd y los procesos netflow, es posible que no tenga permisos de lectura y escritura en el directorio donde nfsen guarda los datos. Para solucionarlo se ha de modificar el fichero $BASEDIR/libexec/Nfcomm.pm y cambiar los permisos del socket para que cualquier usuario pueda leer y escribir

```
debian:~# vi /data1/nfsen/libexec/Nfcomm.pm
```

En la parte del fichero en el que está el condicional:

```
If (1) {
…
my $ok = bind($server, $uaddr);
       if (!$ok) {
```

```
        $Log:ERROR = $!;
        close$server;
        return undef;
}
chmod 0660, $socket_path;
```

Substituir 0660 por 0666

Reiniciar nfsen

```
debian:~# /etc/init.d/nfsen reload
```

A.III. TSA de los enlaces óptimos de RedIRIS

En esta sección se detallan los análisis ARIMA efectuados para cuatro de los enlaces primarios (el resto de los análisis efectuados se encuentran en formato electrónico). Cada análisis presenta dos versiones: la limitada y la extendida. En la versión limitada el análisis está sujeto a la limitación del número de órdenes admitidos, mientras que en la segunda se admiten tantos órdenes como sean necesarios para obtener la máxima calidad de predicción, a costa de la carga computacional relacionada.

A.III.1. Enlace and-can-l

La versión limitada es la incluida en el cuerpo de este mismo trabajo y por esta razón, sólo se incluirá la versión extendida.

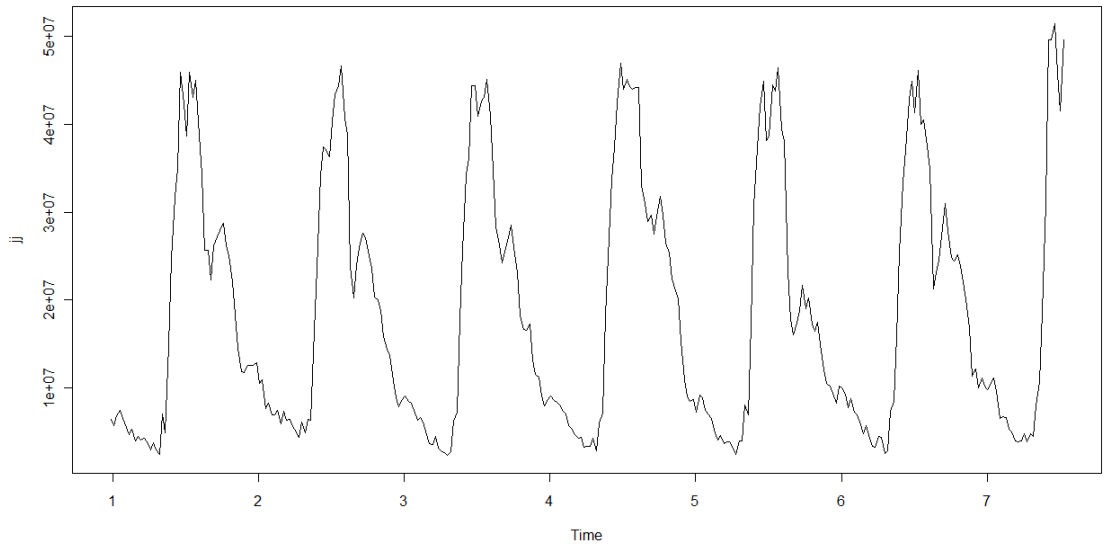
ESPECIFICACIÓN DEL MODELO

- Carga de datos:

jj = ts(scan("c:/Users/Santi/Documents/TFC/Datos DAT/Partidos/and-can-l-lab.dat"), frequency=48)

- Imagen:

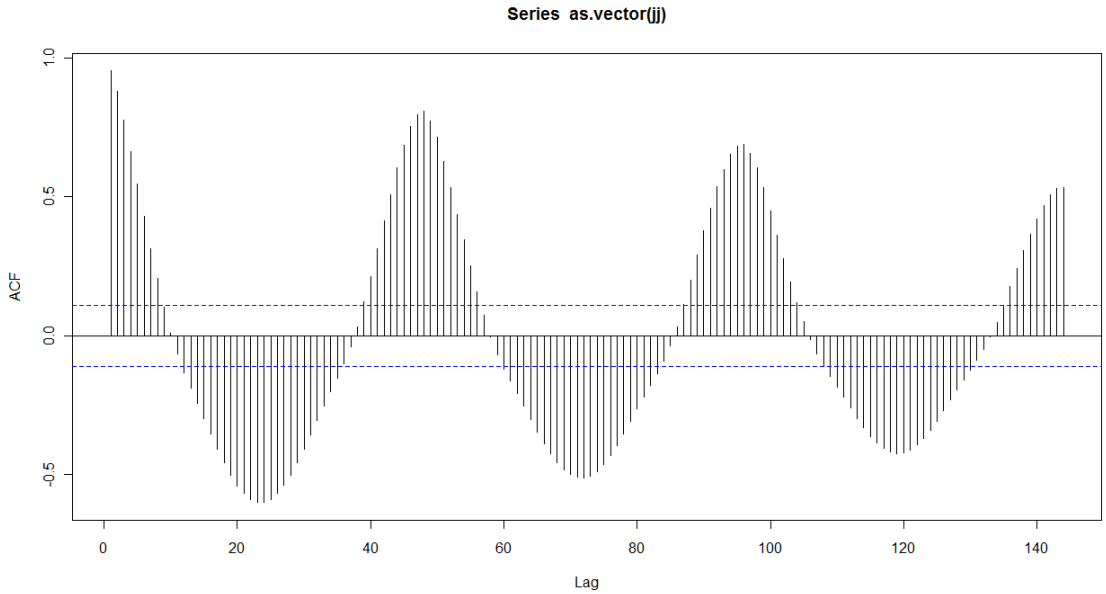
plot(jj)



No se aprecia una tendencia ascendente que haga pensar en una necesaria primera diferenciación (parece ser un modelo estacionario). Sí se observa una clara estacionalidad.

- ACF:

acf(as.vector(jj),lag.max=144)

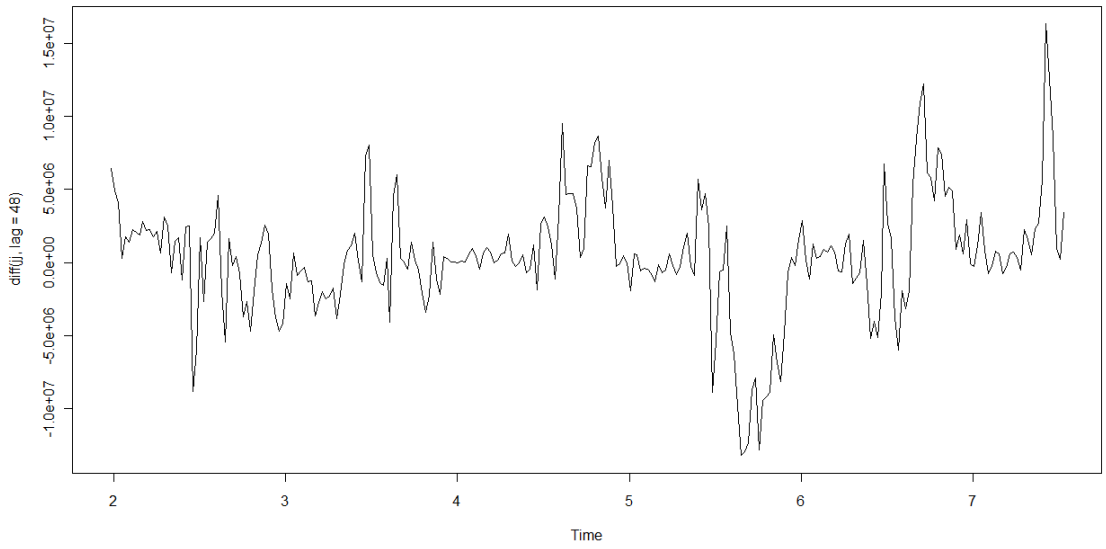


La estacionalidad queda verificada.
A partir de este punto llevaremos a cabo 2 rutas para la especificación del modelo. La primera en la que sólo derivaremos estacionalmente y la segunda en la que además de derivar estacionalmente, aplicaremos la primera derivada sobre el modelo. Partiendo de la necesidad de eliminar la estacionalidad a 48, ¿porqué aplicar también la primera derivada al modelo? La segunda ruta tiene como objetivo ver si hay una no estacionaridad que no se ha observado una vez aplicada la derivada estacional. Tomando estas 2 rutas más probables, esperamos conseguir 2 modelos bastante adecuados, eligiendo después el más prometedor (un compromiso entre simplicidad y adecuación). Se aplicará el siguiente criterio: para este caso, se tomarán TODAS las correlaciones que superen el umbral del 95%.

RUTA 1:

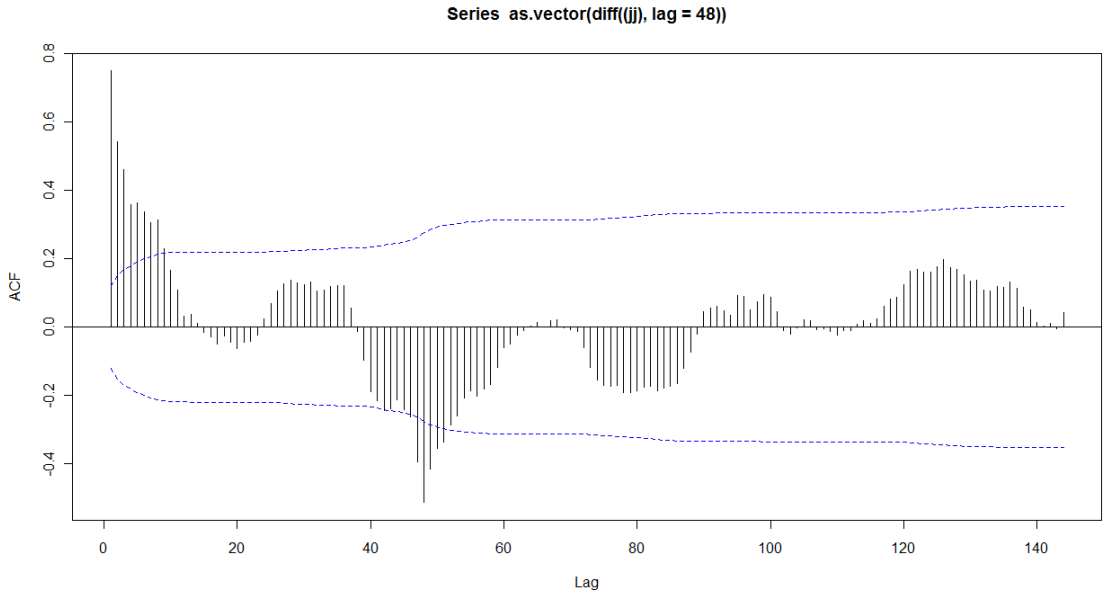
- Derivando estacionalmente con lag = 48 (48 son los datos de un día)

```
plot(diff(jj,lag=48))
```



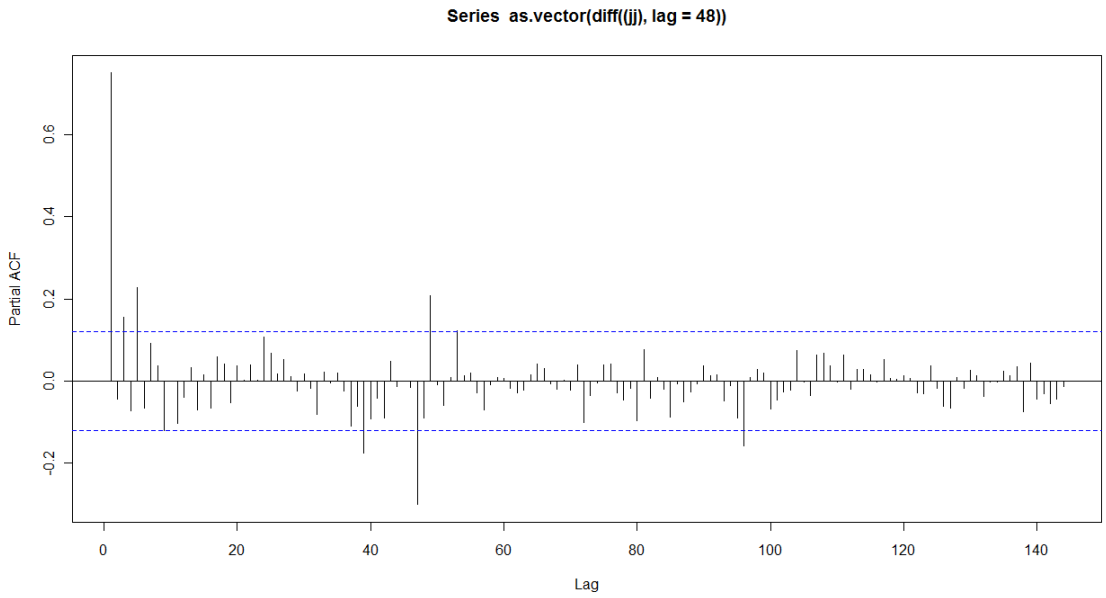
- Aplicando el ACF sobre esta transformación:

acf(as.vector(diff((jj),lag=48)),lag.max=144,ci.type='ma')



- Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff((jj),lag=48)),lag.max=144)

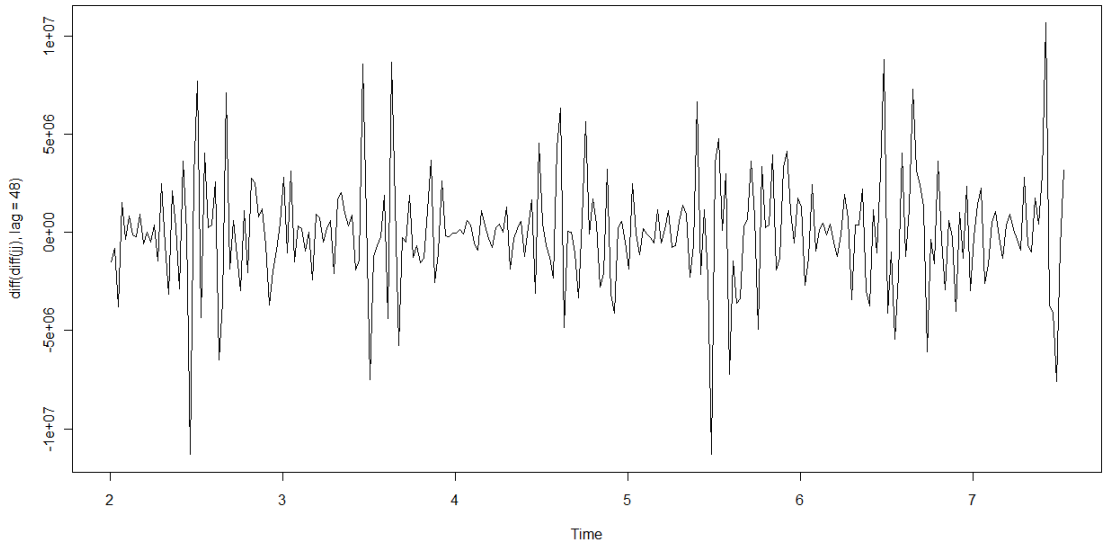


Se puede sugerir que un modelo que incorpore los retrasos 5 y 48 sería adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(5,0,0)x(1,1,0)_{48}$

RUTA 2:

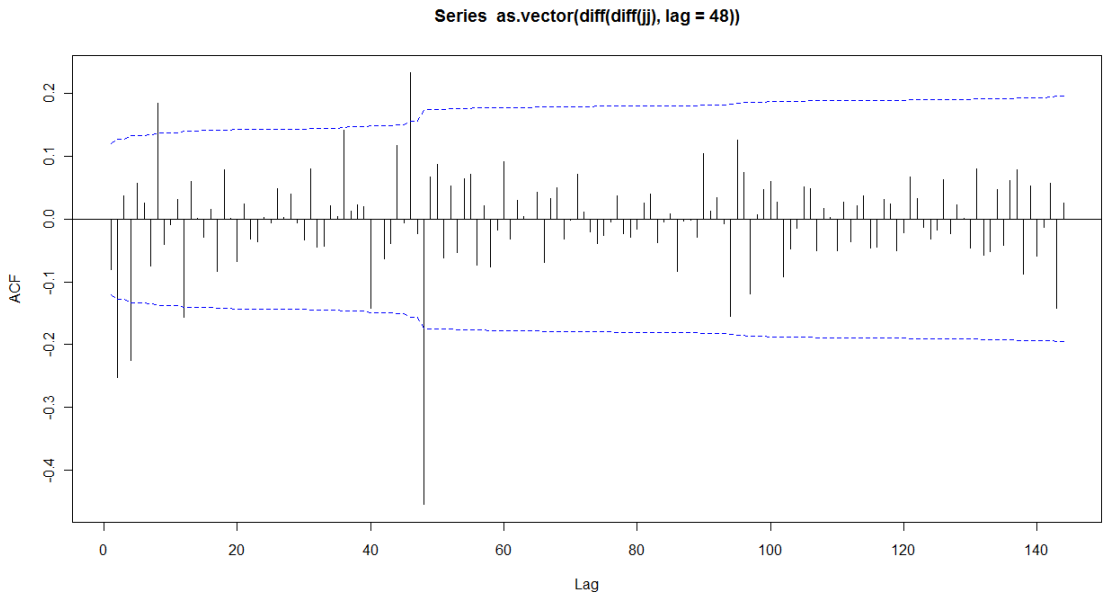
- Derivando estacionalmente con lag = 48 y con la primera derivada

plot(diff(diff(jj),lag=48))



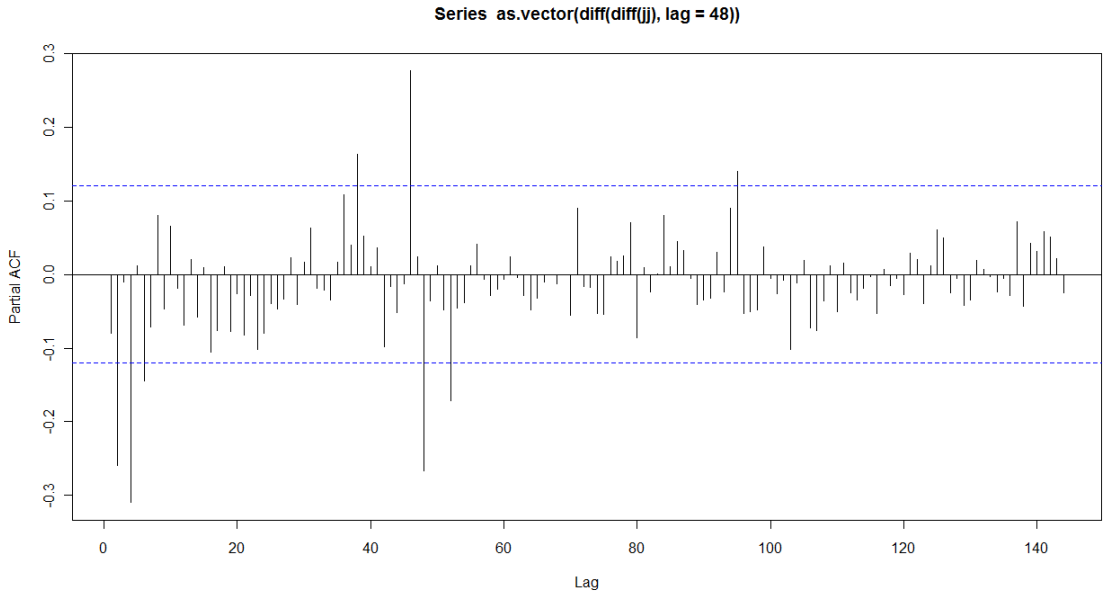
- Aplicando el ACF sobre esta transformación:

acf(as.vector(diff(diff(jj),lag=48)),lag.max=144,ci.type='ma')



- Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff(diff(jj),lag=48)),lag.max=144)



Si se vuelve a observar la gráfica del ACF, se puede sugerir que un modelo que incorpore los retrasos 12 y 48 autocorrelativos seria adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(0,1,12)x(0,1,1)_{48}$

ESTIMACIÓN DE LOS PARÁMETROS

RUTA 1:

```
m1.jj =arima(and_can_l,order=c(5,0,0),seasonal=list(order=c(1,1,0),period=48))
m1.jj
Call:
arima(x = jj, order = c(5, 0, 0), seasonal = list(order = c(1, 1, 0), period = 48))
```

```
Coefficients:
        ar1     ar2     ar3     ar4     ar5     sar1
      0.7768  -0.1198  0.1319  -0.2279  0.2464  -0.6071
s.e.  0.0597   0.0760  0.0758   0.0749  0.0604   0.0522

sigma^2 estimated as 4.225e+12:  log likelihood = -4271.58,  aic = 8555.16
```

RUTA 2:

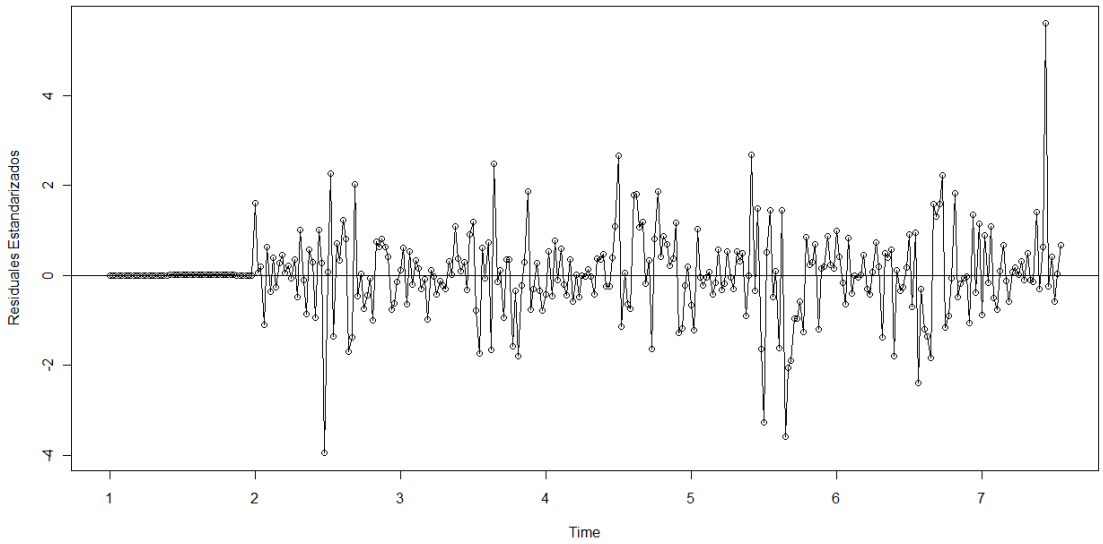
```
m2. and_can_l
=arima(and_can_l,order=c(0,1,12),seasonal=list(order=c(0,1,1),period=48))
m2. and_can_l
Call:
arima(x = jj, order = c(0, 1, 12), seasonal = list(order = c(0, 1, 1), period = 48))

Coefficients:
         ma1     ma2     ma3     ma4     ma5     ma6     ma7     ma8
      -0.1641  -0.3262  -0.0809  -0.1966  -0.0061  0.0952  -0.0845  0.1302
s.e.   0.0616   0.0623   0.0657   0.0660   0.0678  0.0685   0.0711  0.0748
         ma9     ma10     ma11     ma12     sma1
      0.0110  -0.1028  -0.0139  -0.1283  -0.9999
s.e.  0.0668   0.0749   0.0670   0.0670   0.2134

sigma^2 estimated as 3.011e+12:  log likelihood = -4244.89,  aic = 8515.78
```

CHEQUEO DEL DIAGNÓSTICO

A continuación procederemos a comprobar la bondad del afinado de los modelos y para ello, primero hay que observar los residuales estandarizados.

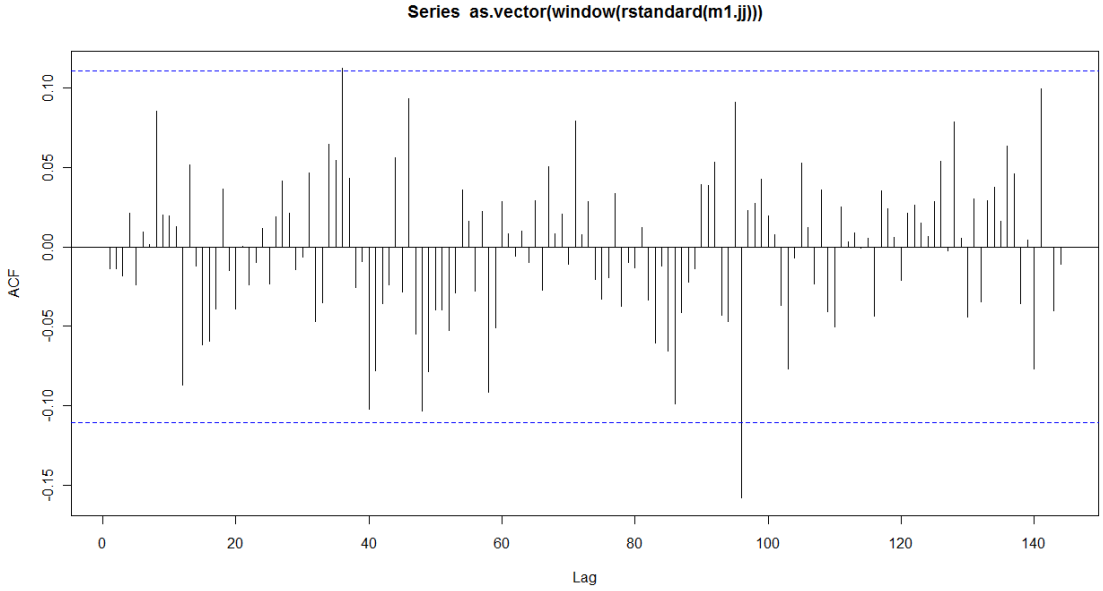
RUTA 1:

```
plot(window(rstandard(m1.                    and_can_l)),ylab='Residuales
Estandarizados',type='o')
abline(h=0)
```



Salvo por unos 3 picos a mediados de la serie y otro al final, el resto de residuales se encuentran acotados entre 2 y -2 aproximadamente. Para precisar más, visualizaremos el ACF de los residuales.

acf(as.vector(window(rstandard(m1. and_can_l))),lag.max=144)



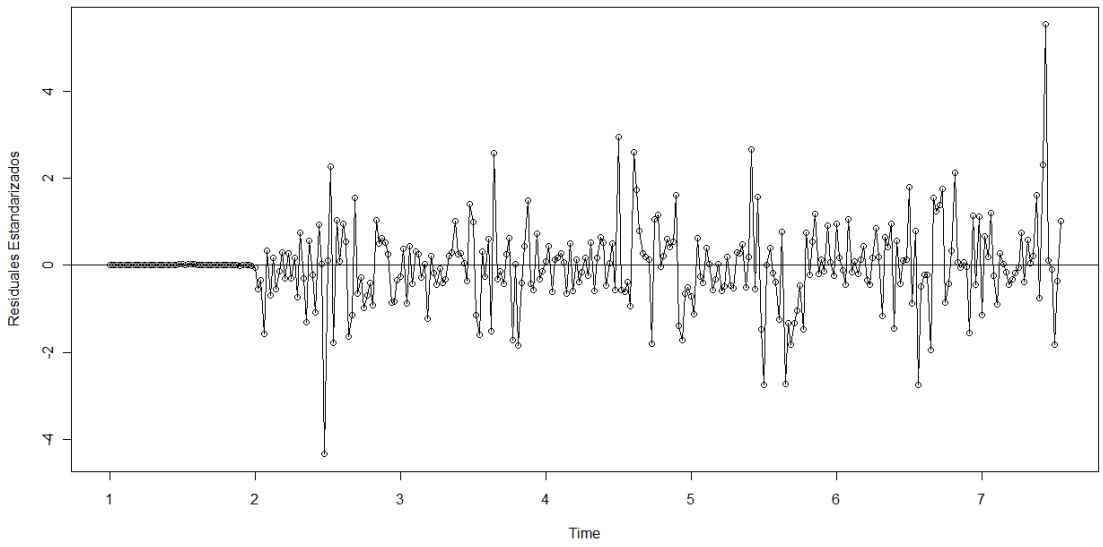
Se aprecia una única correlación “estadísticamente significativa” con un valor del -0,16 en el retraso 96. Aunque es una correlación muy pequeña, podría ser indicativo de algo más, ya que es múltiplo de la frecuencia de datos, 48. Aun así, se puede decir que el modelo ha capturado la esencia de la dependencia de la serie.

Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

var(and_can_l)
[1] 2.006534e+14
SNR = 2.006534e+14 / 4.225e+12 = 47,49

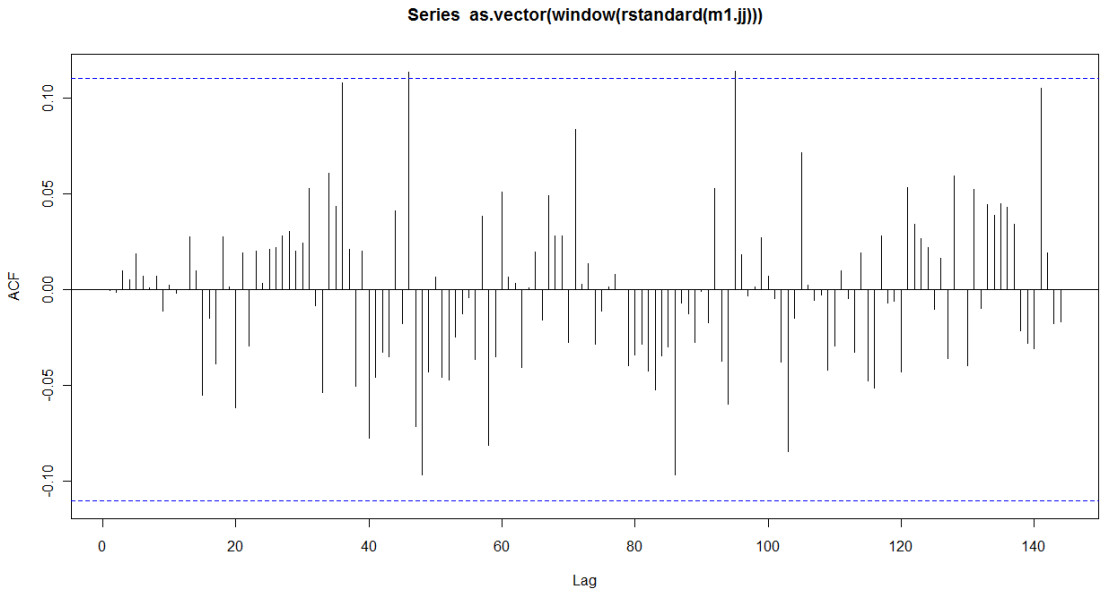
RUTA 2:

```
plot(window(rstandard(m2.and_can_l)),ylab='Residuales
Estandarizados',type='o')
abline(h=0)
```



Los picos que presenta la gráfica son aproximadamente los mismos que para la ruta 1. Para precisar más, visualizaremos el ACF de los residuales.

```
acf(as.vector(window(rstandard(m2. and_can_l))),lag.max=144)
```



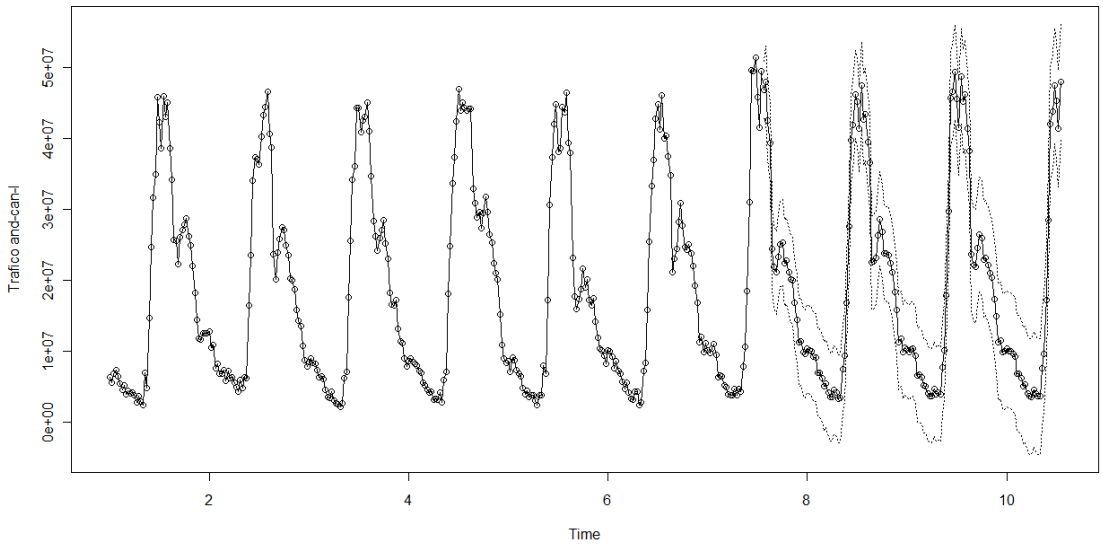
En este caso, sólo dos correlaciones superan escasamente el umbral, con lo que parece que el modelo se ajusta realmente bien a la serie.

Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

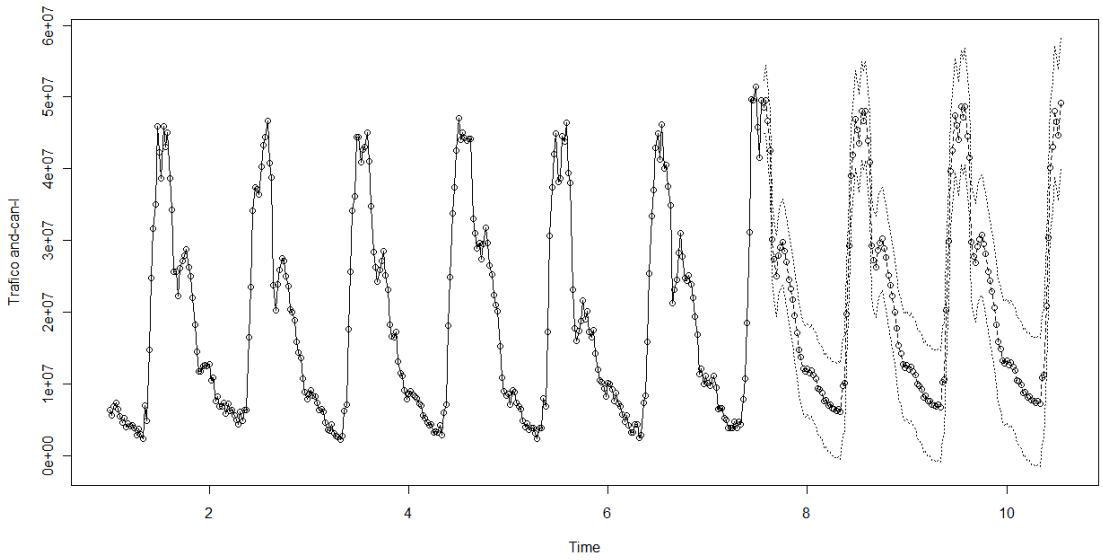
var(and_can_l)
[1] 2.006534e+14
SNR = 2.006534e+14 / 3.011e+12 = 66,64

Si comparamos los resultados de ambas rutas, la ruta 2 presenta una mejor relación señal-ruido y unas correlaciones de los residuales más bajas, lo que parece indicar que el modelo tiene una mejor adecuación a la serie temporal. Esto se corresponde con lo indicado por los atributos AIC retornados para los dos modelos, ya que el menor AIC lo presenta el modelo de la segunda ruta. No obstante, los residuales de la ruta 2 tienen una mayor varianza que los que presenta la ruta 1.

PREDICCIÓN
plot(m1. and_can_l,n.ahead=144,xlab='Time',type='o',ylab='Trafico and-can-l')
ARIMA $(5,0,0)x(1,1,0)_{48}$



plot(m2. and_can_l,n.ahead=144,xlab='Time',type='o',ylab='Trafico and-can-l')
ARIMA $(0,1,12)x(0,1,1)_{48}$



A.III.2. Enlace gal-ast

La versión limitada:

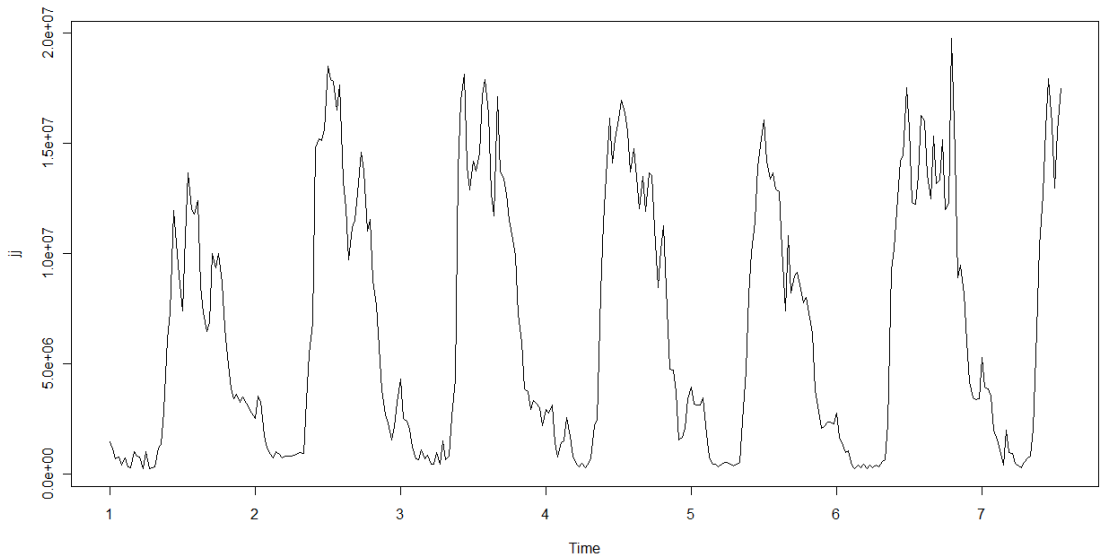
ESPECIFICACIÓN DEL MODELO

- Carga de datos:

jj = ts(scan("c:/Users/Santi/Documents/TFC/Datos DAT/Partidos/gal-ast-lab.dat"), frequency=48)

- Imagen:

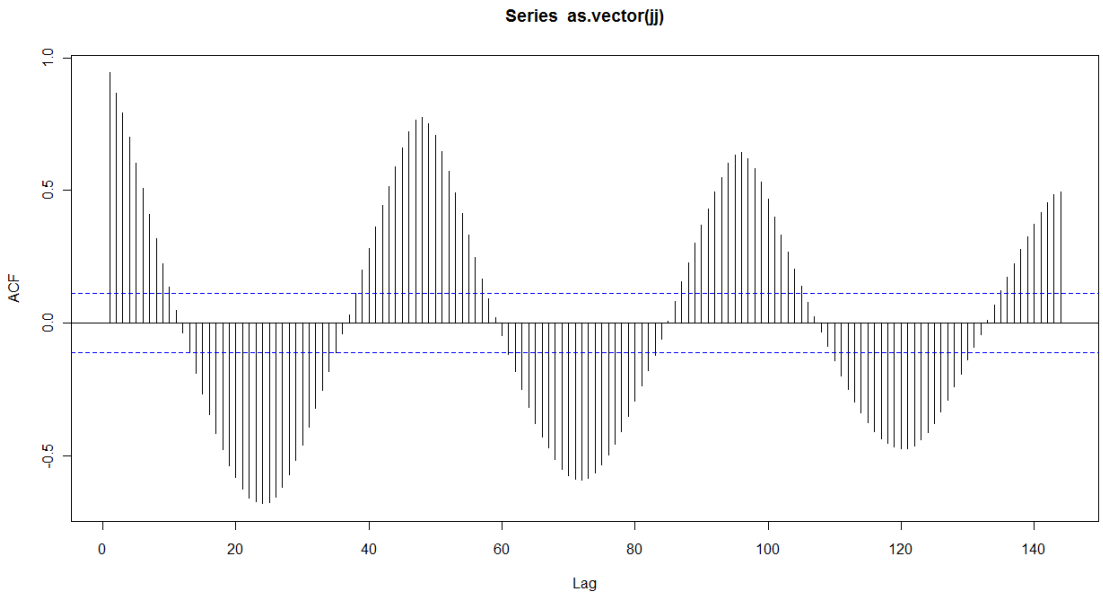
plot(jj)



No se aprecia una tendencia ascendente que haga pensar en una necesaria primera diferenciación (parece ser un modelo estacionario). Sí se observa una clara estacionalidad.

- ACF:

acf(as.vector(jj),lag.max=144)



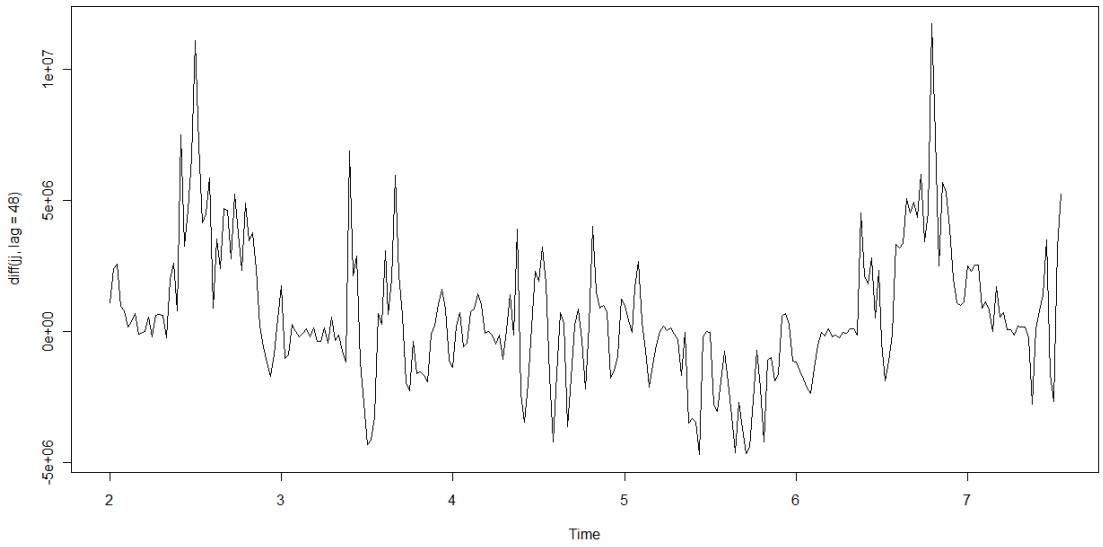
La estacionalidad queda verificada.

A partir de este punto llevaremos a cabo 2 rutas para la especificación del modelo. La primera en la que sólo derivaremos estacionalmente y la segunda en la que además de derivar estacionalmente, aplicaremos la primera derivada sobre el modelo. Partiendo de la necesidad de eliminar la estacionalidad a 48, ¿porqué aplicar también la primera derivada al modelo? La segunda ruta tiene como objetivo ver si hay una no estacionaridad que no se ha observado una vez aplicada la derivada estacional. Tomando estas 2 rutas más probables, esperamos conseguir 2 modelos bastante adecuados, eligiendo después el más prometedor (un compromiso entre simplicidad y adecuación). Se aplicará el siguiente criterio: las correlaciones que se considerarán significativas serán aquellas que superen un factor de 0,2 y con no más de 4 retrasos que superen el umbral del 95%, admitiendo el retraso 48 como necesario a nivel estacional.

RUTA 1:

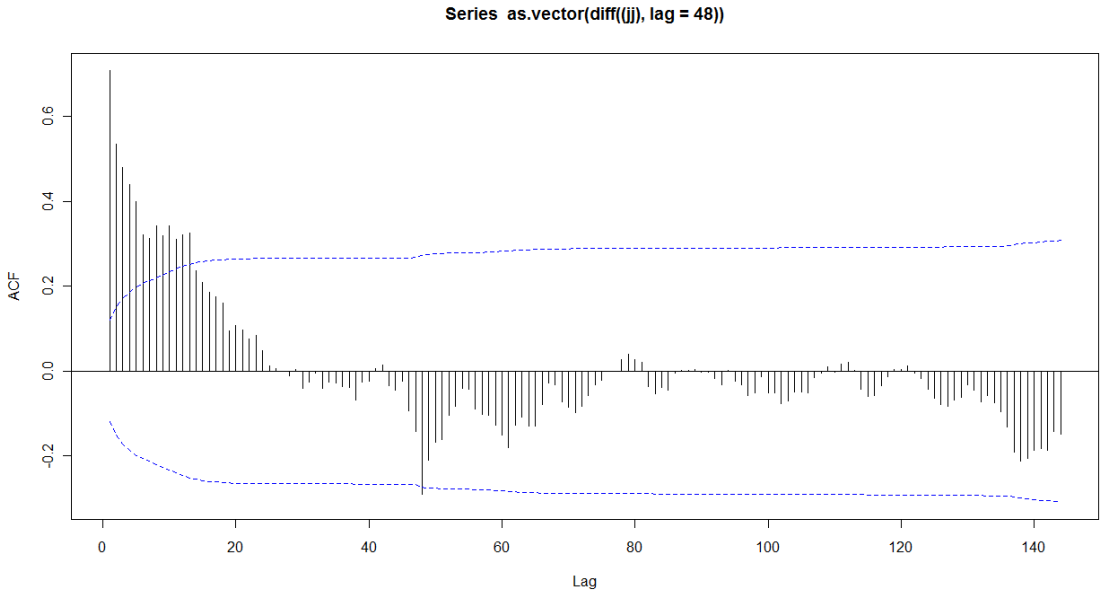
- Derivando estacionalmente con lag = 48 (48 son los datos de un día)

plot(diff(jj,lag=48))



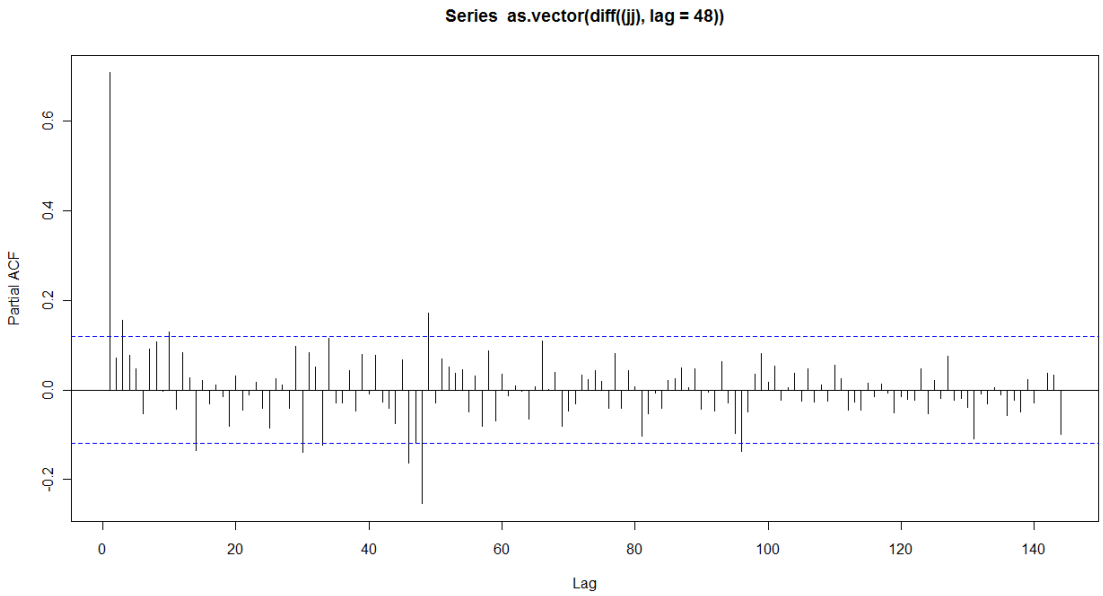
- Aplicando el ACF sobre esta transformación:

acf(as.vector(diff((jj),lag=48)),lag.max=144,ci.type='ma')



- Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff((jj),lag=48)),lag.max=144)

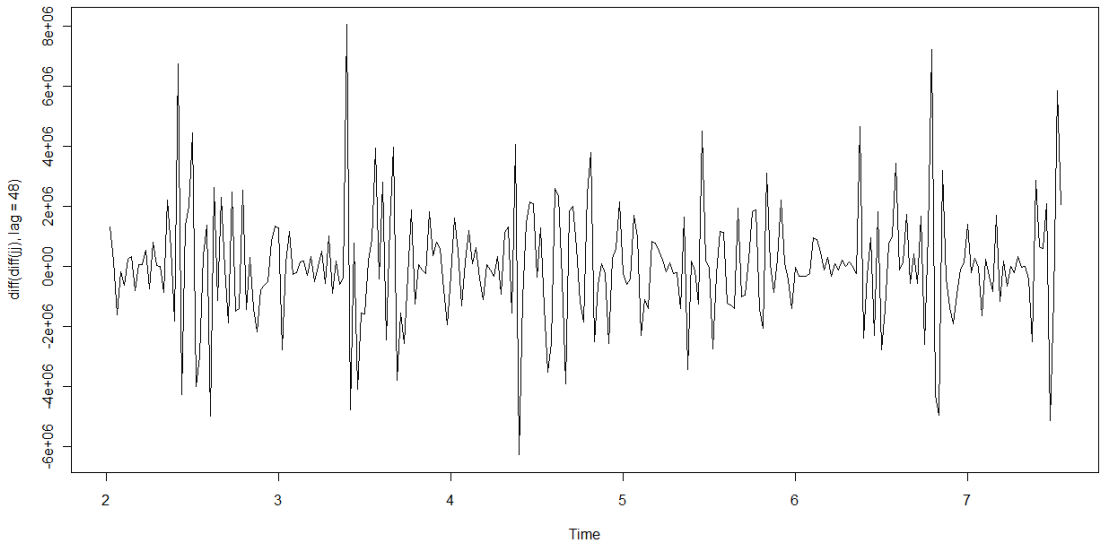


Se puede sugerir que un modelo que incorpore los retrasos 1 y 48 sería adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(1,0,0)x(1,1,0)_{48}$

RUTA 2:

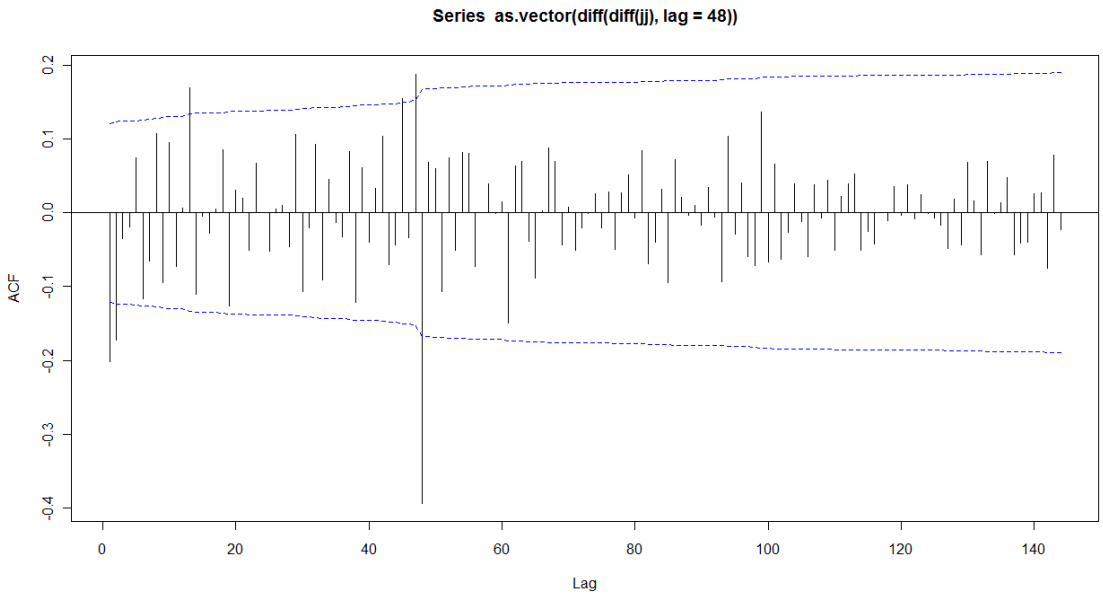
- Derivando estacionalmente con lag = 48 y con la primera derivada

plot(diff(diff(jj),lag=48))



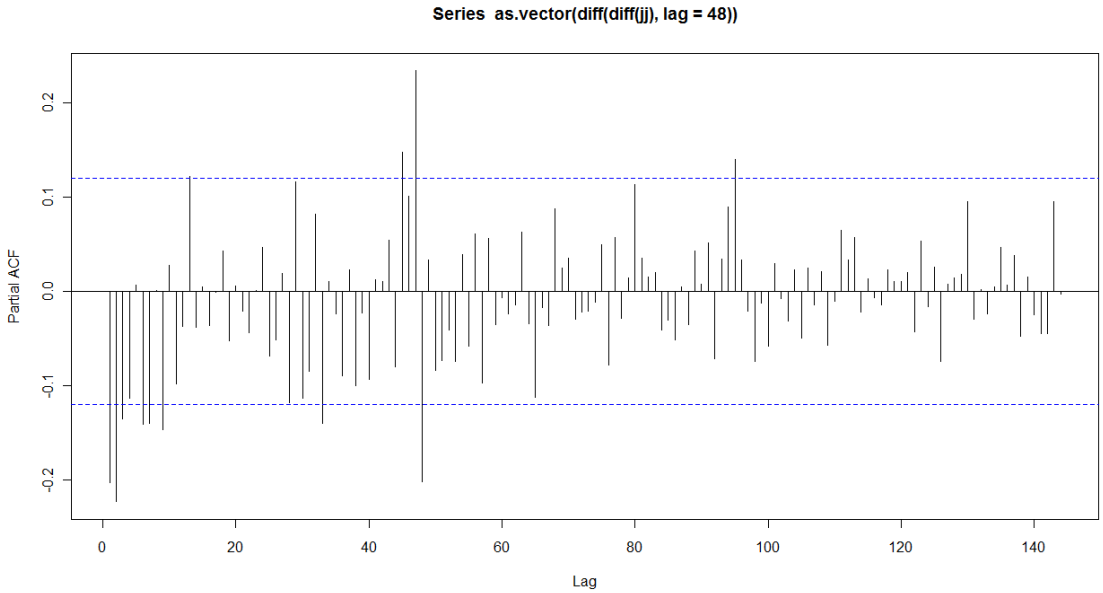
- Aplicando el ACF sobre esta transformación:

acf(as.vector(diff(diff(jj),lag=48)),lag.max=144,ci.type='ma')



- Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff(diff(jj),lag=48)),lag.max=144)



Si se vuelve a observar la gráfica del ACF, se puede sugerir que un modelo que incorpore los retrasos 1 y 48 autocorrelativos seria adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(0,1,1)x(0,1,1)_{48}$

ESTIMACIÓN DE LOS PARÁMETROS

RUTA 1:

```
m1.jj=arima(jj,order=c(1,0,0),seasonal=list(order=c(1,1,0),period=48))
m1.jj
Call:
arima(x = jj, order = c(1, 0, 0), seasonal = list(order = c(1, 1, 0), period = 48))
```

Coefficients:
```
         ar1     sar1
      0.7499  -0.5319
s.e.  0.0408   0.0579
```

sigma^2 estimated as 2.396e+12: log likelihood = -4192.62, aic = 8389.25

RUTA 2:

```
m1.jj=arima(jj,order=c(0,1,1),seasonal=list(order=c(0,1,1),period=48))
m1.jj
```

Call:
arima(x = jj, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 48))

Coefficients:
```
          ma1     sma1
      -0.3702  -0.7126
s.e.   0.0995   0.0946
```

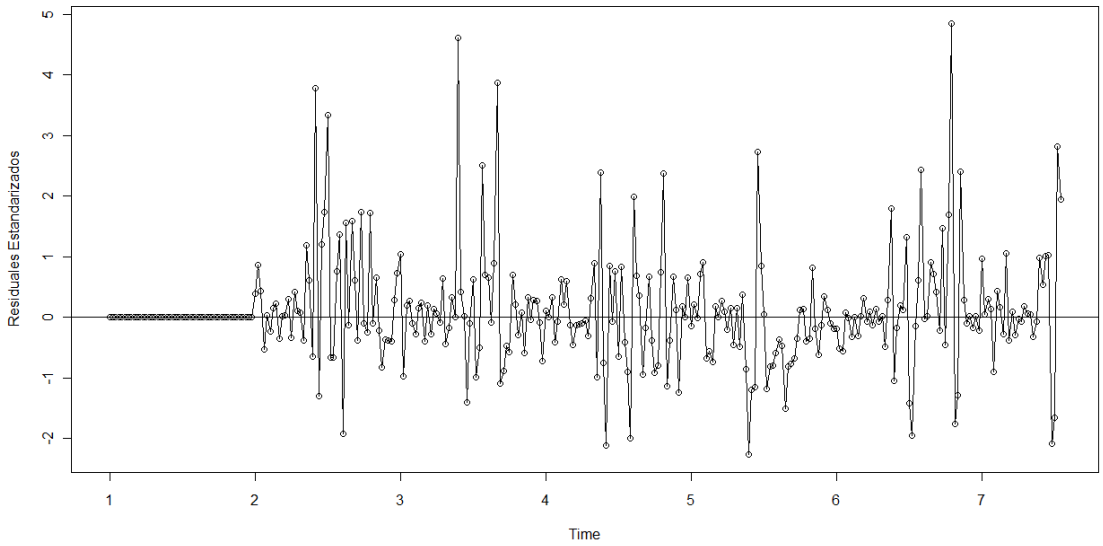
sigma^2 estimated as 2.289e+12: log likelihood = -4179.28, aic = 8362.57

CHEQUEO DEL DIAGNÓSTICO

A continuación procederemos a comprobar la bondad del afinado de los modelos y para ello, primero hay que observar los residuales estandarizados.

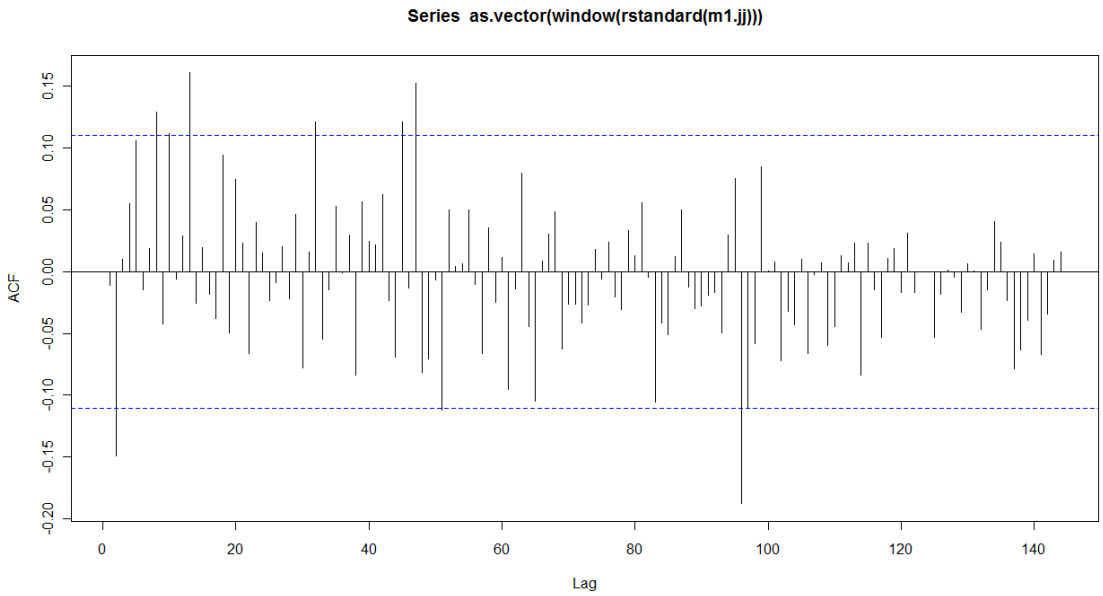
RUTA 1:

```
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)
```



Aunque superiormente hay unos 5 picos que superan ampliamente el 3, el resto de residuales se encuentran acotados entre 2 y -2 aproximadamente. Para precisar más, visualizaremos el ACF de los residuales.

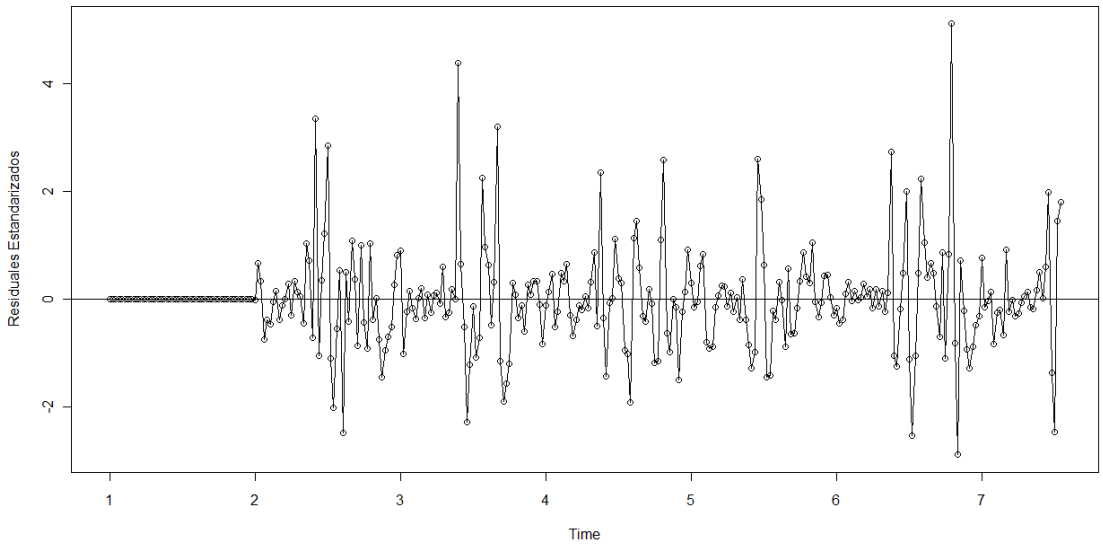
```
acf(as.vector(window(rstandard(m1.jj))),lag.max=144)
```



Se aprecian siete correlaciones “estadísticamente significativas”. La correlación más significativa es la situada en el retraso 96, probablemente debido a que es múltiplo de la frecuencia de datos, 48. Aunque presenta un valor de correlación del entorno del -0,19, es un valor muy bajo y se puede decir que el modelo ha capturado la esencia de la dependencia de la serie.

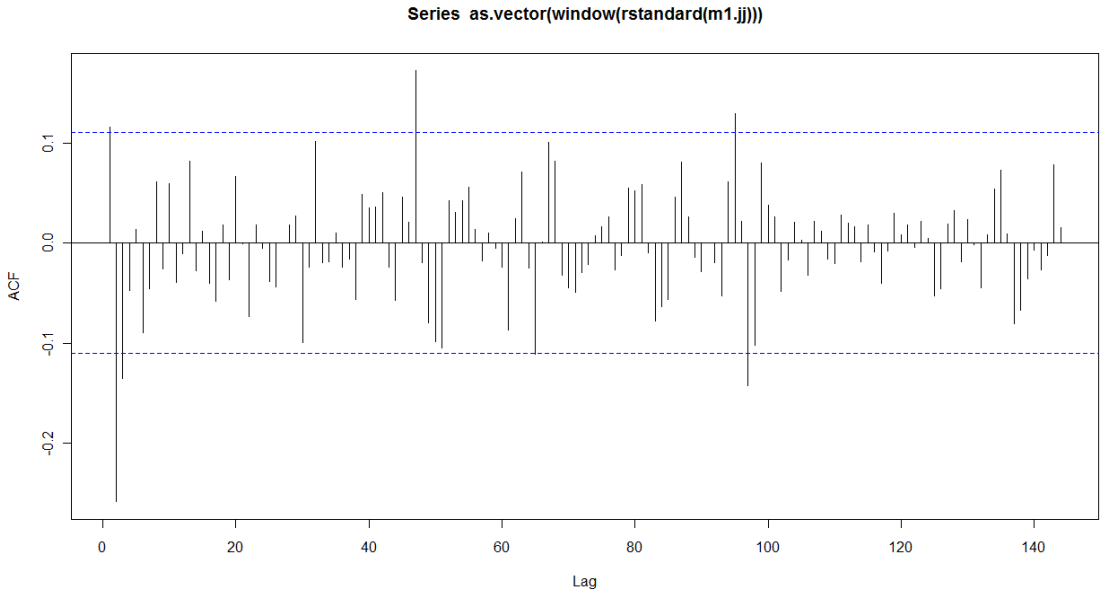
Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.
var(jj)
[1] 3.347823e+13
SNR = 3.347823e+13 / 2.396e+12 = 13,97

RUTA 2:
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)



Los picos que presenta la gráfica son aproximadamente los mismos que para la ruta 1 aunque su varianza se ha reducido. Para precisar más, visualizaremos el ACF de los residuales.

acf(as.vector(window(rstandard(m1.jj))),lag.max=144)



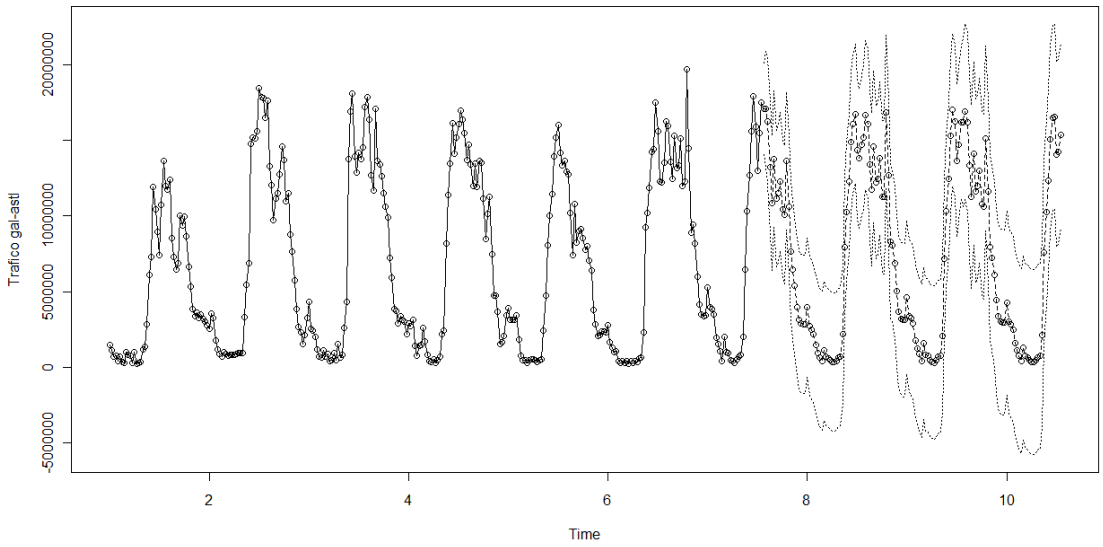
Hay 5 correlaciones que superan el umbral. De ellas, la situada en el retraso 2 es la más elevada y presenta un valor de unos -0,25, notablemente más elevada que para el caso de la ruta 1.

Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

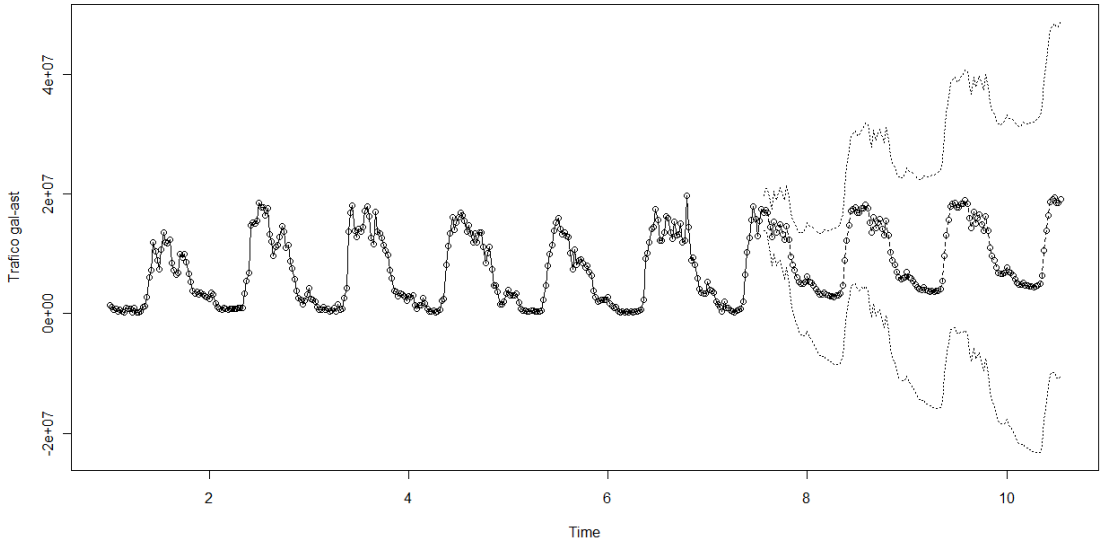
var(jj)
[1] 3.347823e+13
SNR = 3.347823e+13 / 2.289e+12 = 14,63

Si comparamos los resultados de ambas rutas, la ruta 2 presenta una mejor relación señal-ruido y unas correlaciones de los residuales aproximadamente iguales, lo que parece indicar que el modelo tiene una mejor adecuación a la serie temporal. Esto se corresponde con lo indicado por los atributos AIC retornados para los dos modelos, ya que el menor AIC lo presenta el modelo de la segunda ruta. No obstante, las predicciones para de la ruta 2 presentan una mayor varianza que los de la ruta 1.

PREDICCIÓN
plot(m1.jj,n.ahead=144,xlab='Time',type='o',ylab='Trafico gal-astl')
ARIMA $(1,0,0)x(1,1,0)_{48}$



ARIMA $(0,1,1)x(0,1,1)_{48}$



La versión extendida:

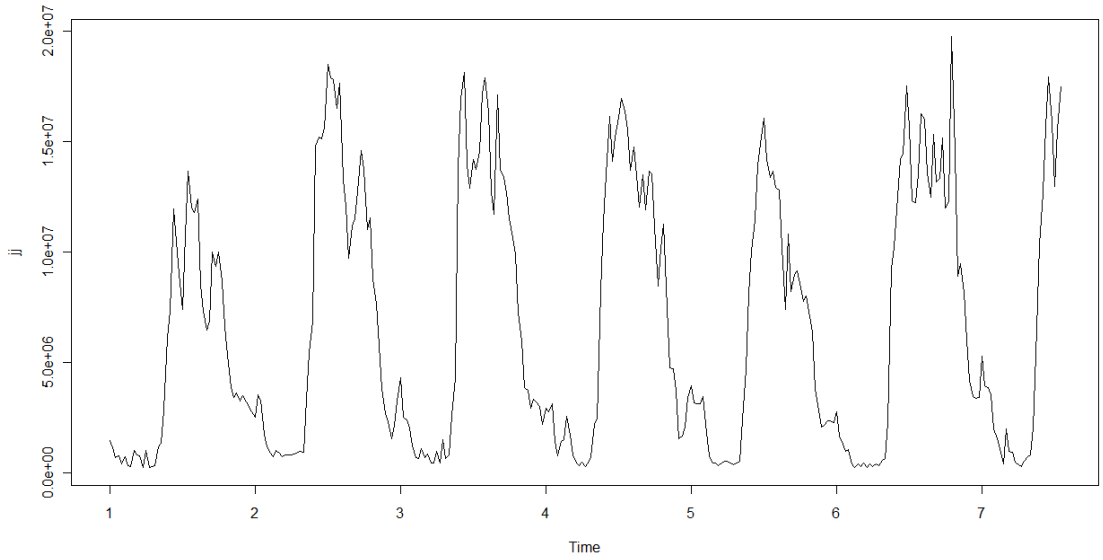
ESPECIFICACIÓN DEL MODELO

- Carga de datos:

jj = ts(scan("c:/Users/Santi/Documents/TFC/Datos DAT/Partidos/gal-ast-lab.dat"), frequency=48)

- Imagen:

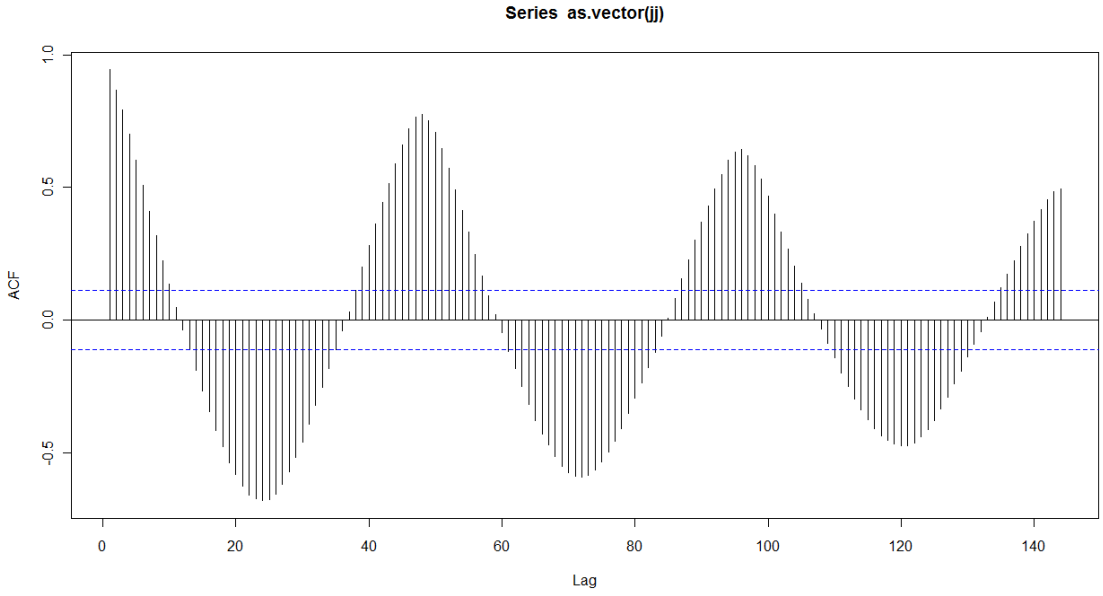
plot(jj)



No se aprecia una tendencia ascendente que haga pensar en una necesaria primera diferenciación (parece ser un modelo estacionario). Sí se observa una clara estacionalidad.

- ACF:

acf(as.vector(jj),lag.max=144)



La estacionalidad queda verificada.

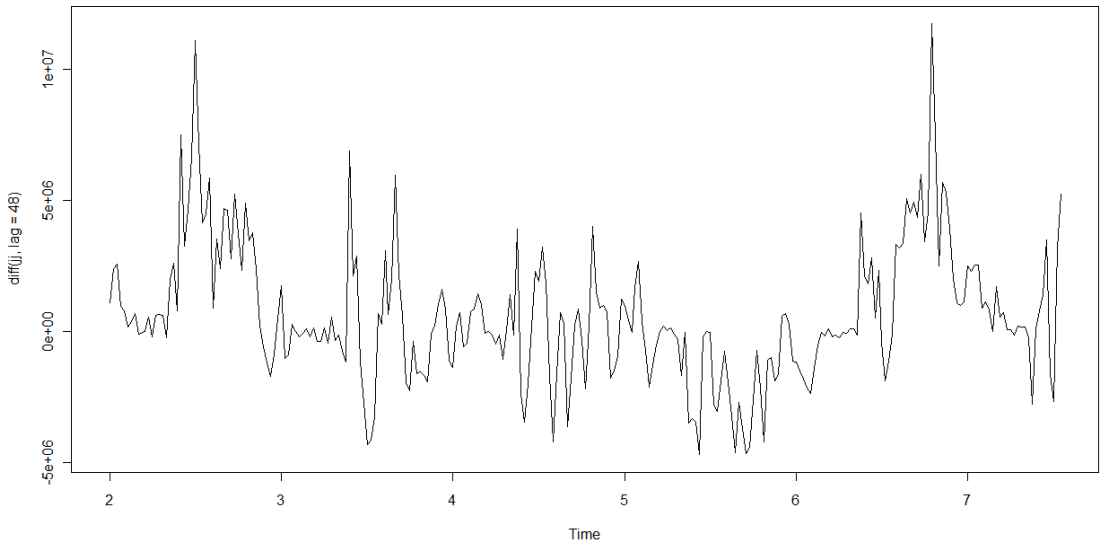
A partir de este punto llevaremos a cabo 2 rutas para la especificación del modelo. La primera en la que sólo derivaremos estacionalmente y la segunda en la que además de derivar estacionalmente, aplicaremos la primera derivada sobre el modelo. Partiendo de la necesidad de eliminar la estacionalidad a 48, ¿porqué aplicar también la primera derivada al modelo? La segunda ruta tiene como objetivo ver si hay una no estacionaridad que no se ha observado una vez aplicada la derivada estacional. Tomando estas 2 rutas más probables,

esperamos conseguir 2 modelos bastante adecuados, eligiendo después el más prometedor (un compromiso entre simplicidad y adecuación). Se aplicará el siguiente criterio: para este caso, se tomarán TODAS las correlaciones que superen el umbral del 95%.

RUTA 1:

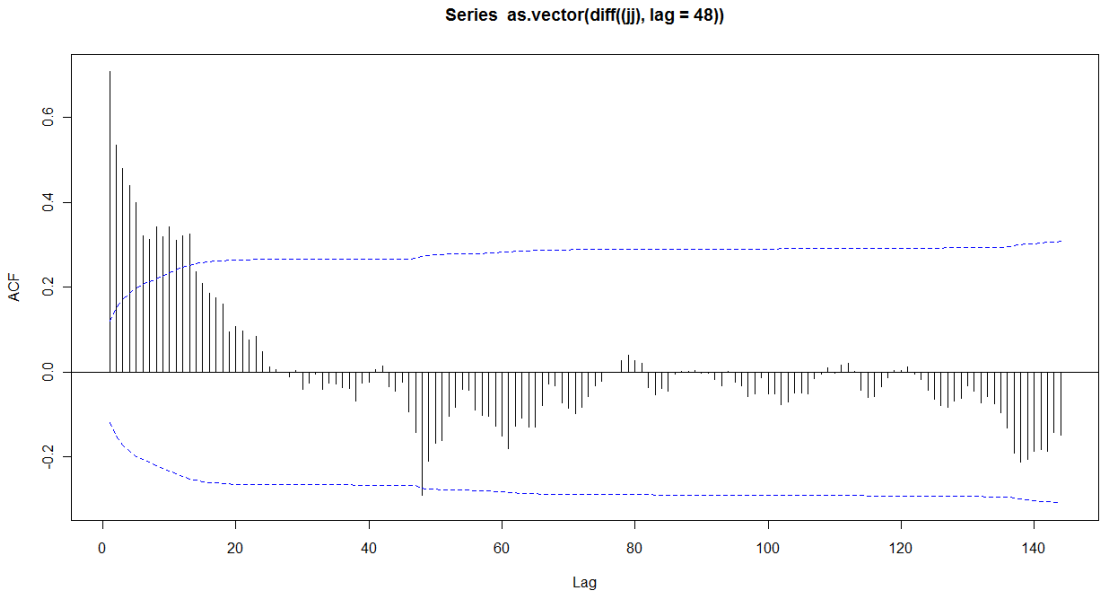
- Derivando estacionalmente con lag = 48 (48 son los datos de un día)

plot(diff(jj,lag=48))



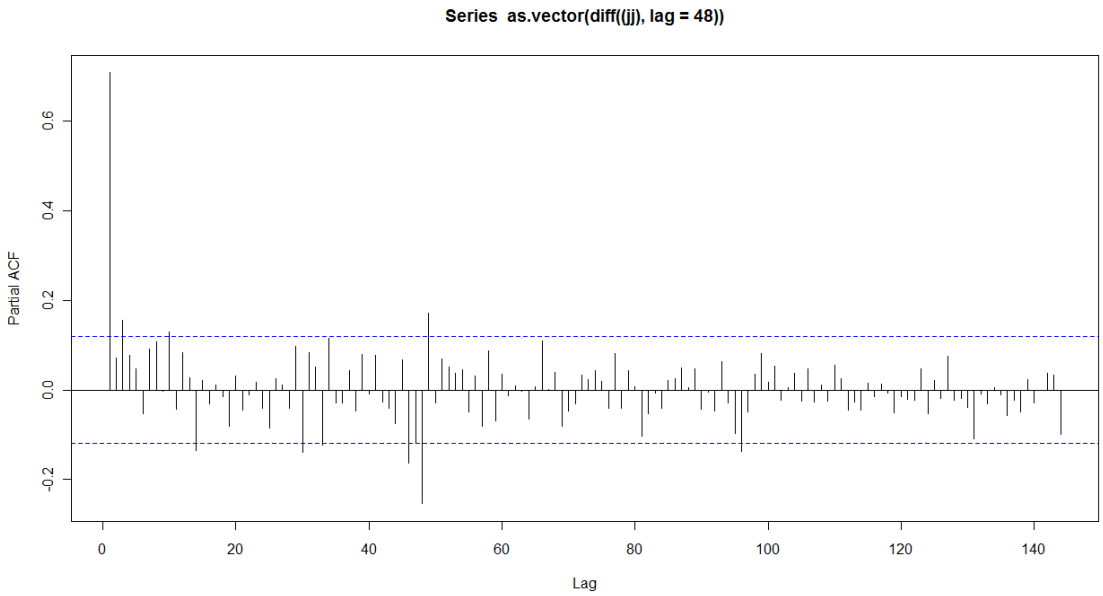
- Aplicando el ACF sobre esta transformación:

acf(as.vector(diff((jj),lag=48)),lag.max=144,ci.type='ma')



- Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff((jj),lag=48)),lag.max=144)

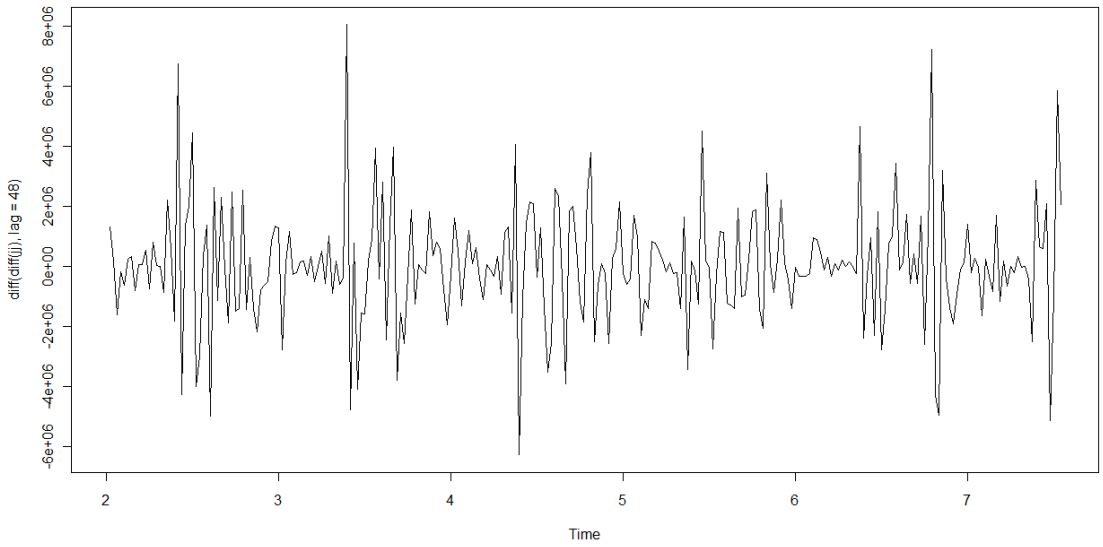


Se puede sugerir que un modelo que incorpore los retrasos 14 y 48 sería adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(14,0,0)x(1,1,0)_{48}$

RUTA 2:

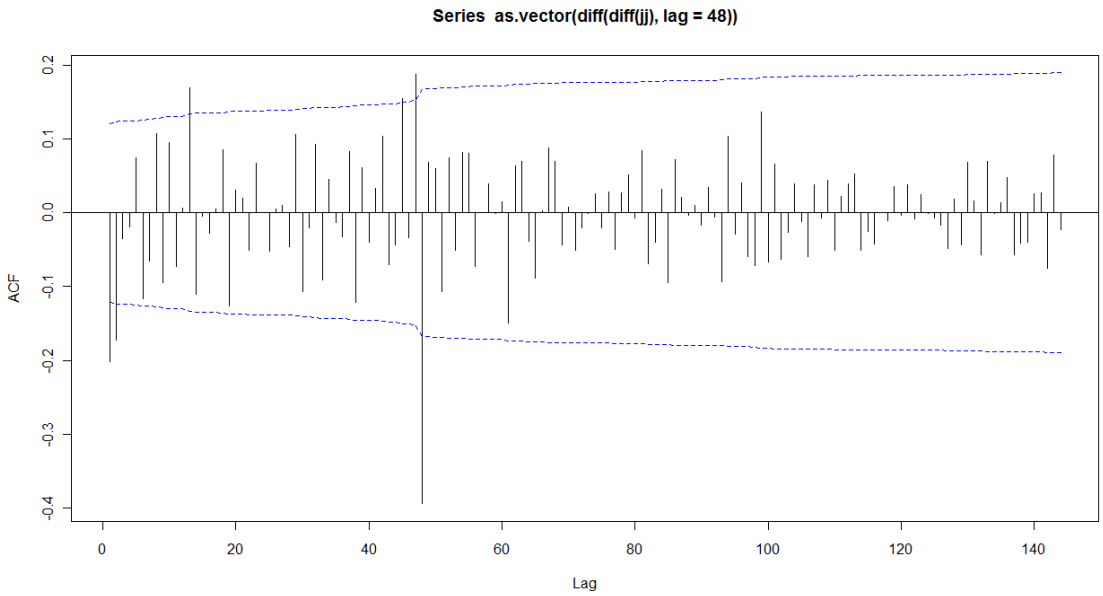
- Derivando estacionalmente con lag = 48 y con la primera derivada

plot(diff(diff(jj),lag=48))



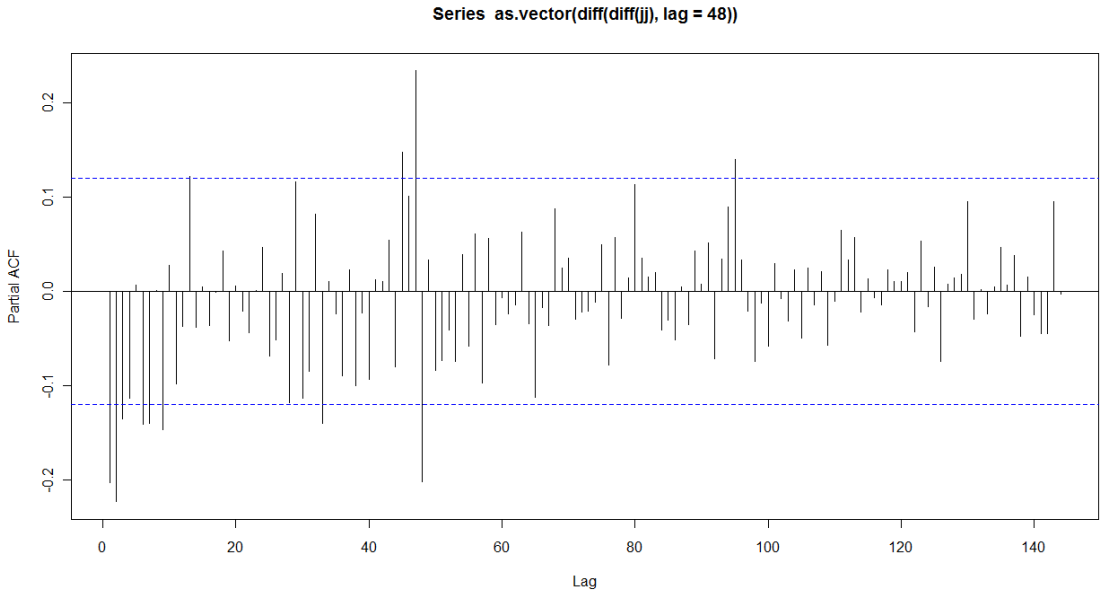
- Aplicando el ACF sobre esta transformación:

acf(as.vector(diff(diff(jj),lag=48)),lag.max=144,ci.type='ma')



- Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff(diff(jj),lag=48)),lag.max=144)



Si se vuelve a observar la gráfica del ACF, se puede sugerir que un modelo que incorpore los retrasos 13 y 48 autocorrelativos seria adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(0,1,13)x(0,1,1)_{48}$

ESTIMACIÓN DE LOS PARÁMETROS

RUTA 1:

```
m1.jj=arima(jj,order=c(14,0,0),seasonal=list(order=c(1,1,0),period=48))
m1.jj
Call:
arima(x = jj, order = c(14, 0, 0), seasonal = list(order = c(1, 1, 0), period = 48))
```

Coefficients:

	ar1	ar2	ar3	ar4	ar5	ar6	ar7	ar8	ar9
	0.7382	-0.1651	0.1193	0.0589	0.0531	-0.0480	0.0343	0.0984	-0.1433
s.e.	0.0616	0.0770	0.0774	0.0772	0.0764	0.0761	0.0760	0.0755	0.0756

	ar10	ar11	ar12	ar13	ar14	sar1
	0.1850	-0.1089	0.0514	0.1064	-0.1231	-0.5427
s.e.	0.0761	0.0771	0.0769	0.0762	0.0616	0.0585

sigma^2 estimated as 2.134e+12: log likelihood = -4177.98, aic = 8385.95

RUTA 2:

```
m1.jj=arima(jj,order=c(0,1,13),seasonal=list(order=c(0,1,1),period=48))
m1.jj
```

Call:
arima(x = jj, order = c(0, 1, 13), seasonal = list(order = c(0, 1, 1), period = 48))

Coefficients:

	ma1	ma2	ma3	ma4	ma5	ma6	ma7	ma8
	-0.2426	-0.3795	-0.0647	-0.0067	0.0080	-0.0519	-0.0655	0.1097
s.e.	0.0621	0.0650	0.0684	0.0679	0.0723	0.0729	0.0793	0.0740

	ma9	ma10	ma11	ma12	ma13	sma1
	-0.0481	0.0155	-0.0334	-0.0995	0.0537	-0.8541
s.e.	0.0693	0.0671	0.0866	0.0794	0.0693	0.1790

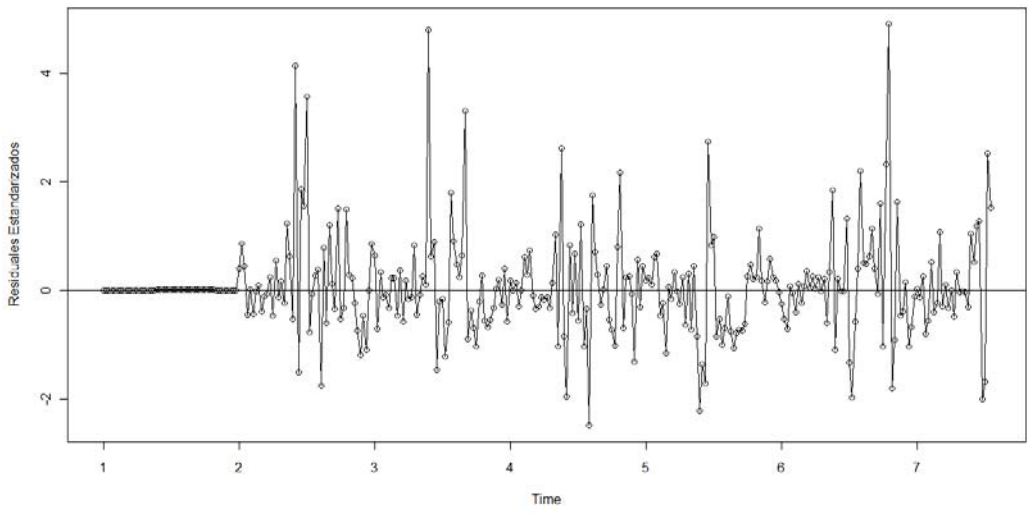
sigma^2 estimated as 1.779e+12: log likelihood = -4157.63, aic = 8343.26

CHEQUEO DEL DIAGNÓSTICO

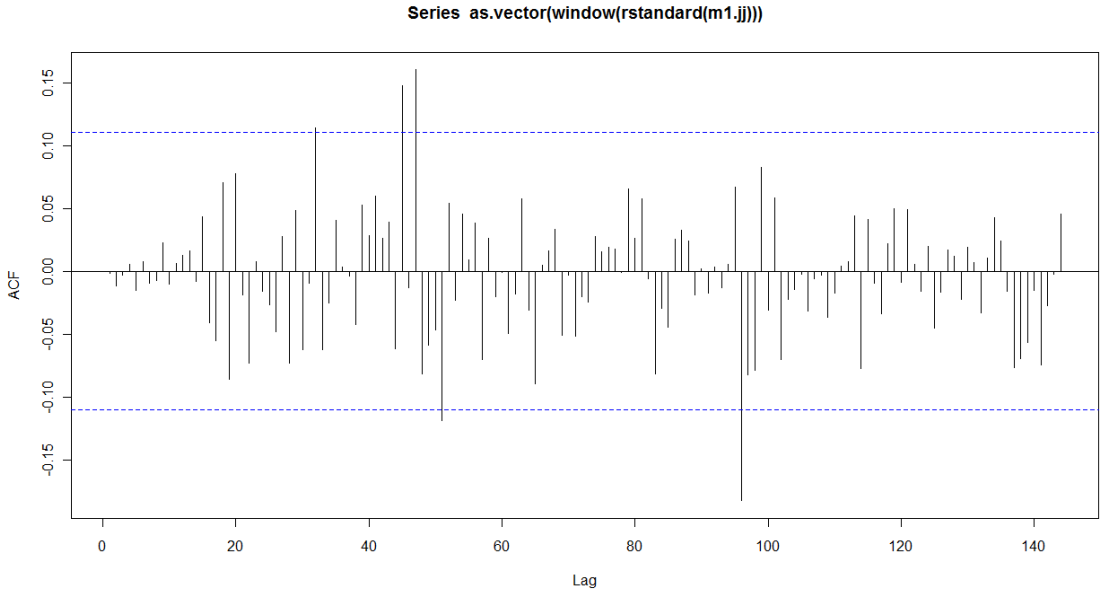
A continuación procederemos a comprobar la bondad del afinado de los modelos y para ello, primero hay que observar los residuales estandarizados.

RUTA 1:

```
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)
```



Salvo por unos 5 picos a mediados de la serie que rondan el 4, el resto de residuales se encuentran acotados entre 2 y -2 aproximadamente. Para precisar más, visualizaremos el ACF de los residuales.
acf(as.vector(window(rstandard(m1.jj))),lag.max=144)

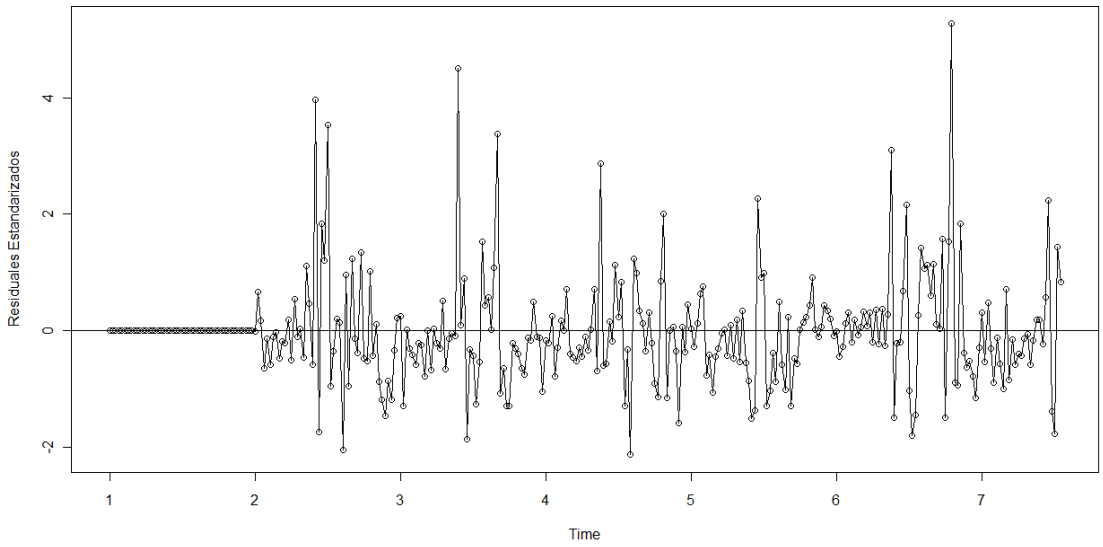


Se aprecian dos correlaciones “estadísticamente significativas” con un valor de 0,15 o superior, y una a unos -0,17 en el retraso 96. Aunque es una correlación muy pequeña, podría ser indicativo de algo más, ya que es múltiplo de la frecuencia de datos, 48. Aun así, se puede decir que el modelo ha capturado la esencia de la dependencia de la serie.

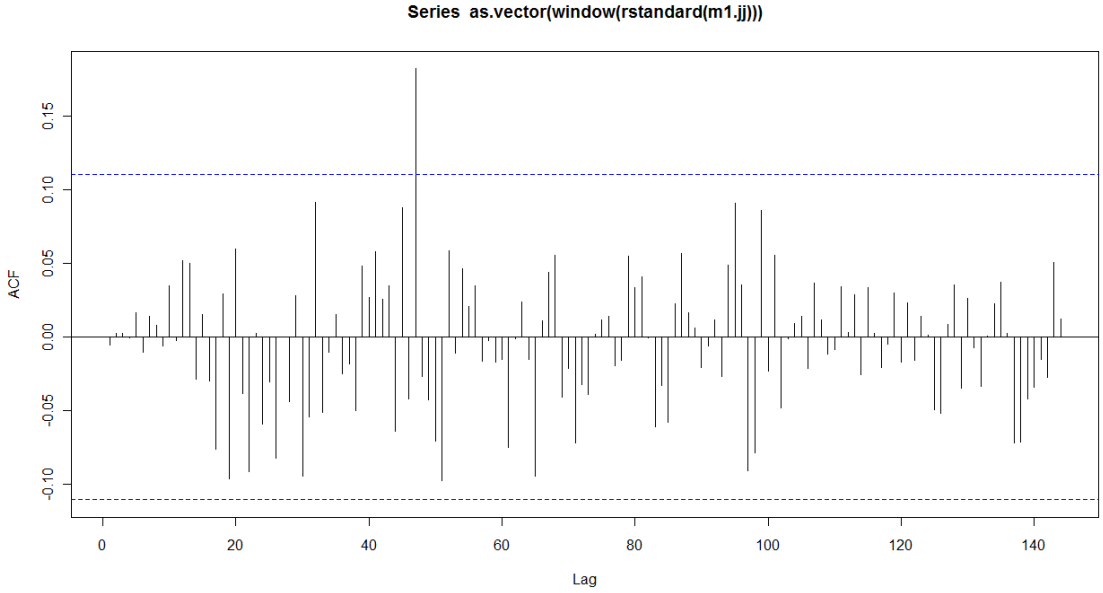
Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

var(jj)
[1] 3.347823e+13
SNR = 3.347823e+13 / 2.134e+12 = 15,69

RUTA 2:
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)



Los picos que presenta la gráfica son aproximadamente los mismos que para la ruta 1. Para precisar más, visualizaremos el ACF de los residuales.
acf(as.vector(window(rstandard(m1.jj))),lag.max=144)



En este caso, sólo hay una correlación que supera el umbral, con un valor de 0,15, con lo que parece que el modelo se ajusta a la serie muy bien.
Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

var(jj)
[1] 3.347823e+13
SNR = 3.347823e+13 / 1.779e+12 = 18,82

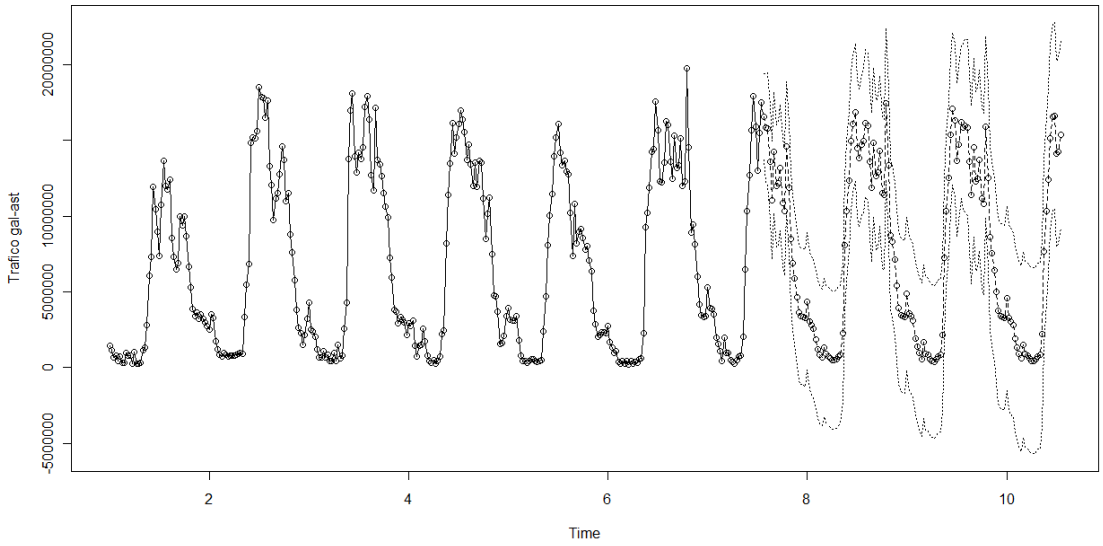
Si comparamos los resultados de ambas rutas, la ruta 2 presenta una mejor relación señal-ruido y unas correlaciones de los residuales más bajas y menos numerosas, lo que parece indicar que el modelo tiene una mejor adecuación a

la serie temporal. Esto se corresponde con lo indicado por los atributos AIC retornados para los dos modelos, ya que el menor AIC lo presenta el modelo de la segunda ruta. La varianza de los residuales parece ser aproximadamente la misma para ambas rutas.

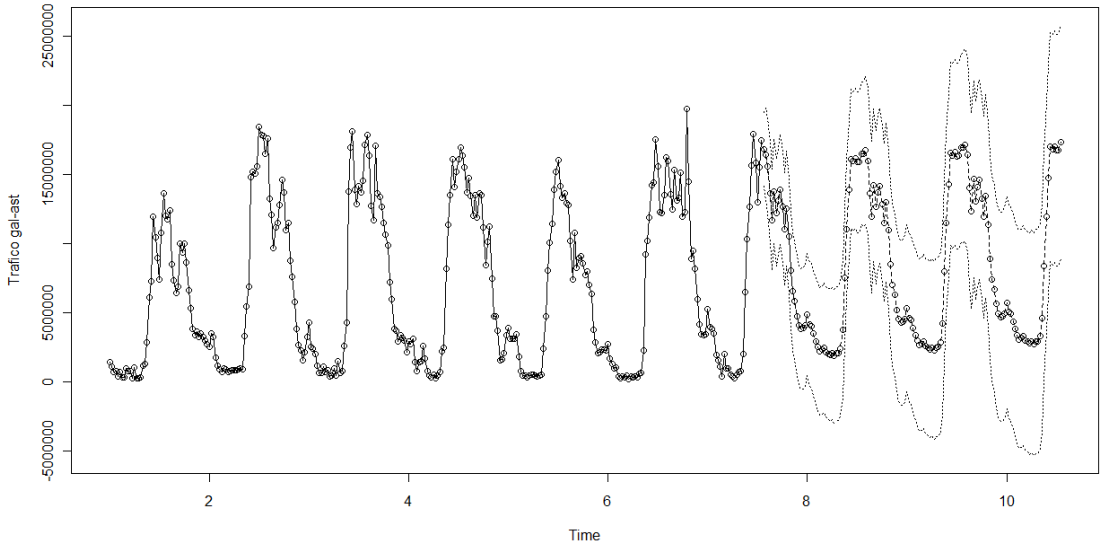
PREDICCIÓN

plot(m1.jj,n.ahead=144,xlab='Time',type='o',ylab='Trafico gal-ast')

ARIMA $(14,0,0)x(1,1,0)_{48}$



ARIMA $(0,1,13)x(0,1,1)_{48}$



A.III.3. Enlace mad-nac1

La versión limitada:

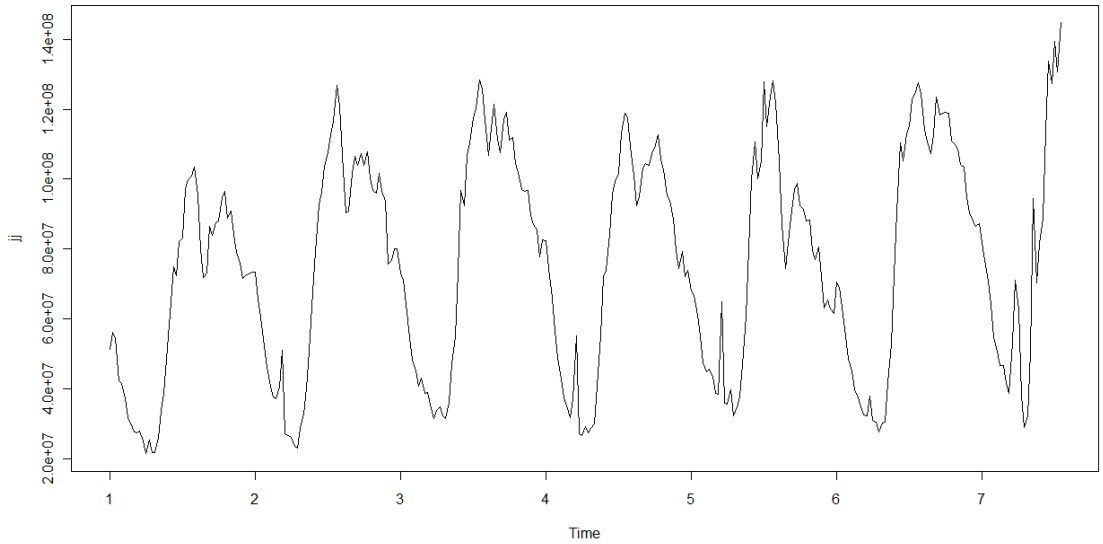
ESPECIFICACIÓN DEL MODELO

- Carga de datos:

jj = ts(scan("c:/Users/Santi/Documents/TFC/Datos DAT/Partidos/mad-nac1-lab.dat"), frequency=48)

- Imagen:

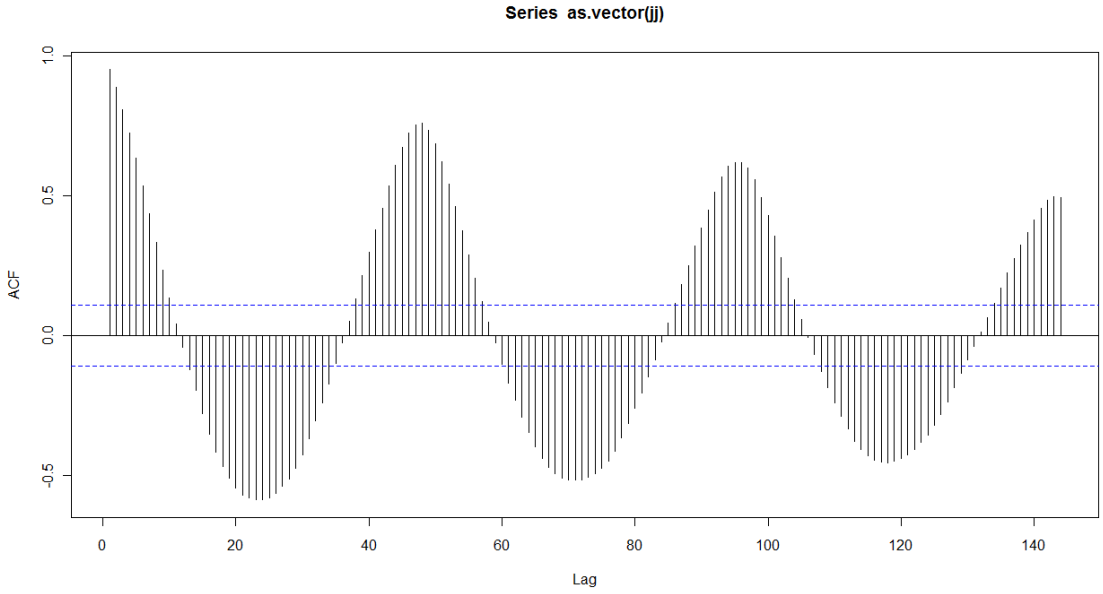
plot(jj)



No se aprecia una tendencia ascendente que haga pensar en una necesaria primera diferenciación (parece ser un modelo estacionario). Sí se observa una clara estacionalidad.

- ACF:

acf(as.vector(jj),lag.max=144)



La estacionalidad queda verificada.

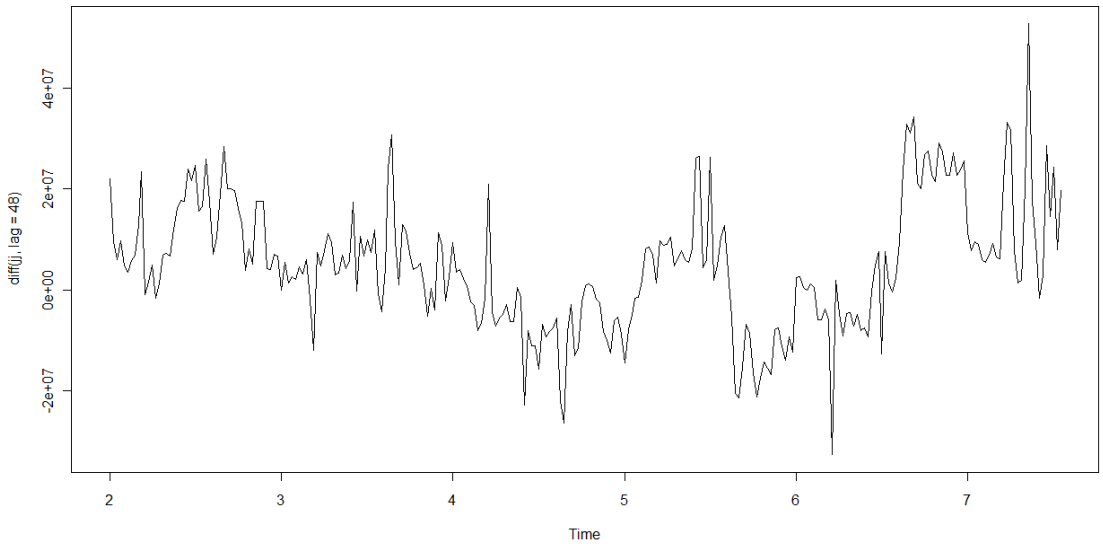
A partir de este punto llevaremos a cabo 2 rutas para la especificación del modelo. La primera en la que sólo derivaremos estacionalmente y la segunda en la que además de derivar estacionalmente, aplicaremos la primera derivada sobre el modelo. Partiendo de la necesidad de eliminar la estacionalidad a 48,

¿porqué aplicar también la primera derivada al modelo? La segunda ruta tiene como objetivo ver si hay una no estacionaridad que no se ha observado una vez aplicada la derivada estacional. Tomando estas 2 rutas más probables, esperamos conseguir 2 modelos bastante adecuados, eligiendo después el más prometedor (un compromiso entre simplicidad y adecuación). Se aplicará el siguiente criterio: las correlaciones que se considerarán significativas serán aquellas que superen un factor de 0,2 y con no más de 4 retrasos que superen el umbral del 95%, admitiendo el retraso 48 como necesario a nivel estacional.

RUTA 1:

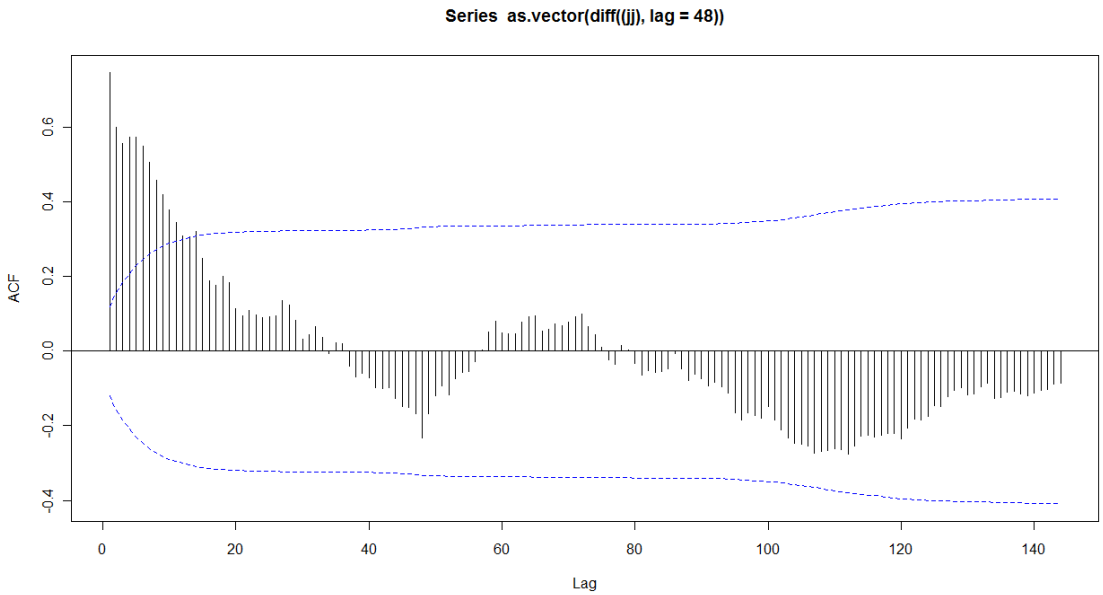
- Derivando estacionalmente con lag = 48 (48 son los datos de un día)

plot(diff(jj,lag=48))



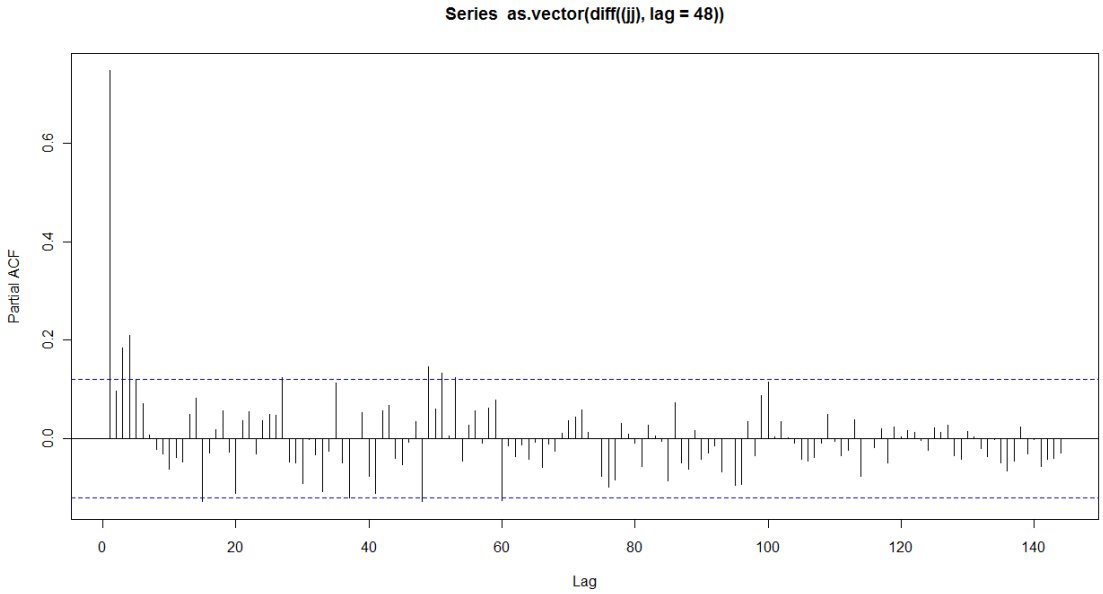
o Aplicando el ACF sobre esta transformación:

acf(as.vector(diff((jj),lag=48)),lag.max=144,ci.type='ma')



o Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff((jj),lag=48)),lag.max=144)

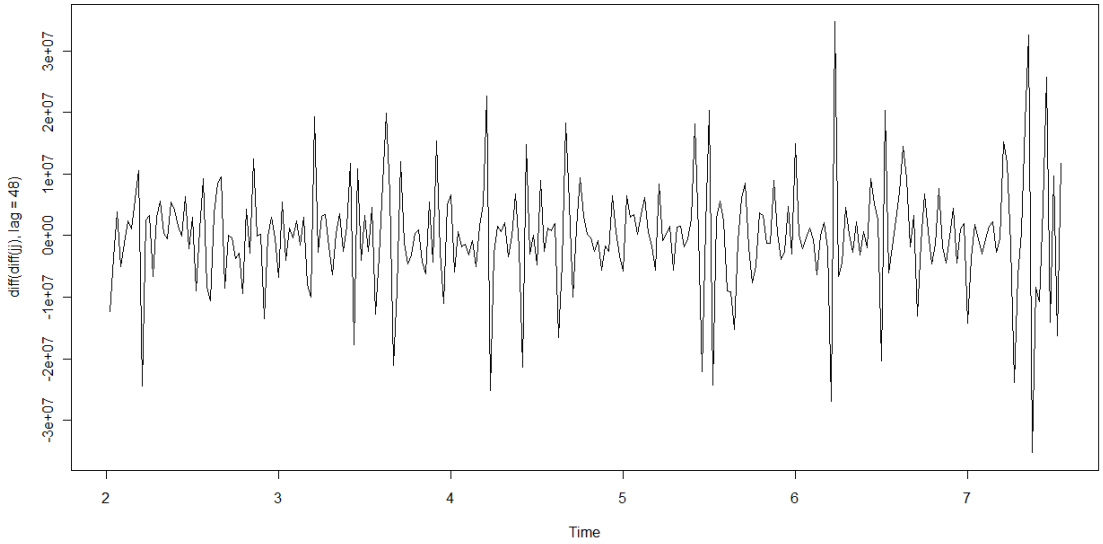


Se puede sugerir que un modelo que incorpore los retrasos 4 y 48 sería adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(4,0,0)x(1,1,0)_{48}$

RUTA 2:

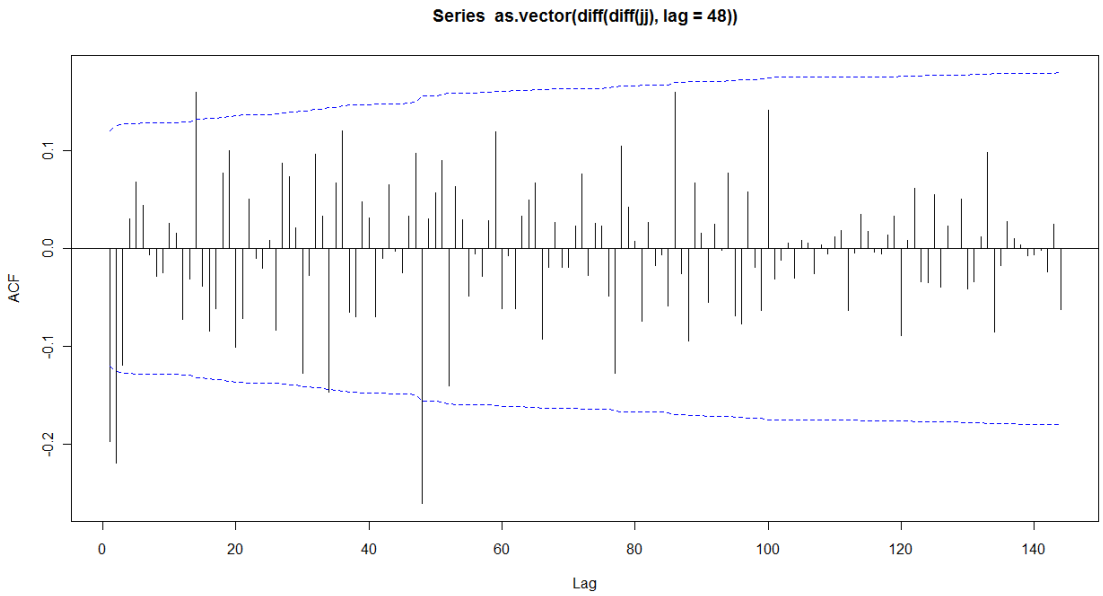
- Derivando estacionalmente con lag = 48 y con la primera derivada

plot(diff(diff(jj),lag=48))



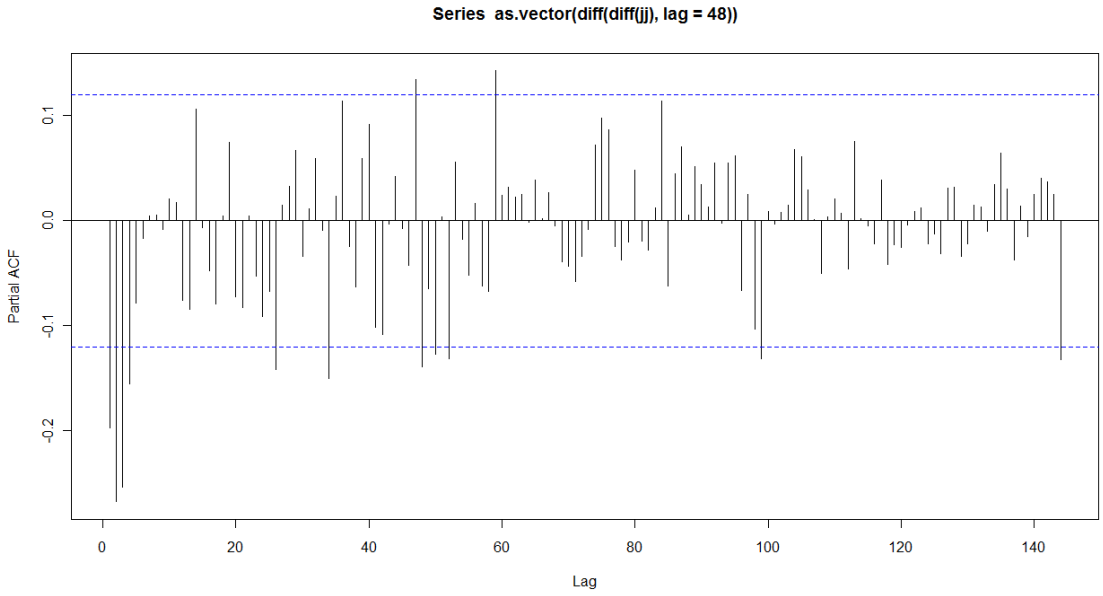
o Aplicando el ACF sobre esta transformación:

acf(as.vector(diff(diff(jj),lag=48)),lag.max=144,ci.type='ma')



- Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff(diff(jj),lag=48)),lag.max=144)



Si se vuelve a observar la gráfica del ACF, se puede sugerir que un modelo que incorpore los retrasos 2 y 48 autocorrelativos seria adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(0,1,2)x(0,1,1)_{48}$

ESTIMACIÓN DE LOS PARÁMETROS

RUTA 1:

```
m1.jj=arima(jj,order=c(4,0,0),seasonal=list(order=c(1,1,0),period=48))
m1.jj
Call:
arima(x = jj, order = c(4, 0, 0), seasonal = list(order = c(1, 1, 0), period = 48))

Coefficients:
```

```
        ar1      ar2     ar3     ar4     sar1
      0.6415  -0.0422  0.0751  0.2151  -0.3464
s.e.  0.0601   0.0718  0.0721  0.0603   0.0656

sigma^2 estimated as 5.671e+13:  log likelihood = -4610.4,  aic = 9230.79
```

RUTA 2:

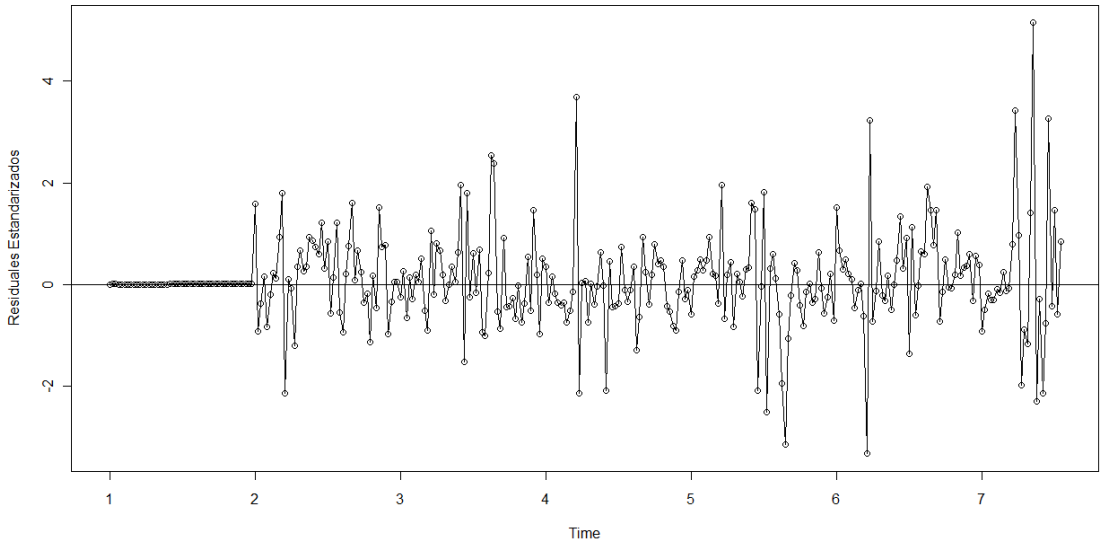
```
m1.jj=arima(jj,order=c(0,1,2),seasonal=list(order=c(0,1,1),period=48))
m1.jj

Call:
arima(x = jj, order = c(0, 1, 2), seasonal = list(order = c(0, 1, 1), period = 48))

Coefficients:
         ma1      ma2     sma1
      -0.3994  -0.2850  -0.7029
s.e.   0.0578   0.0554   0.0979

sigma^2 estimated as 4.693e+13:  log likelihood = -4580.6,  aic = 9167.19
```

CHEQUEO DEL DIAGNÓSTICO

A continuación procederemos a comprobar la bondad del afinado de los modelos y para ello, primero hay que observar los residuales estandarizados.

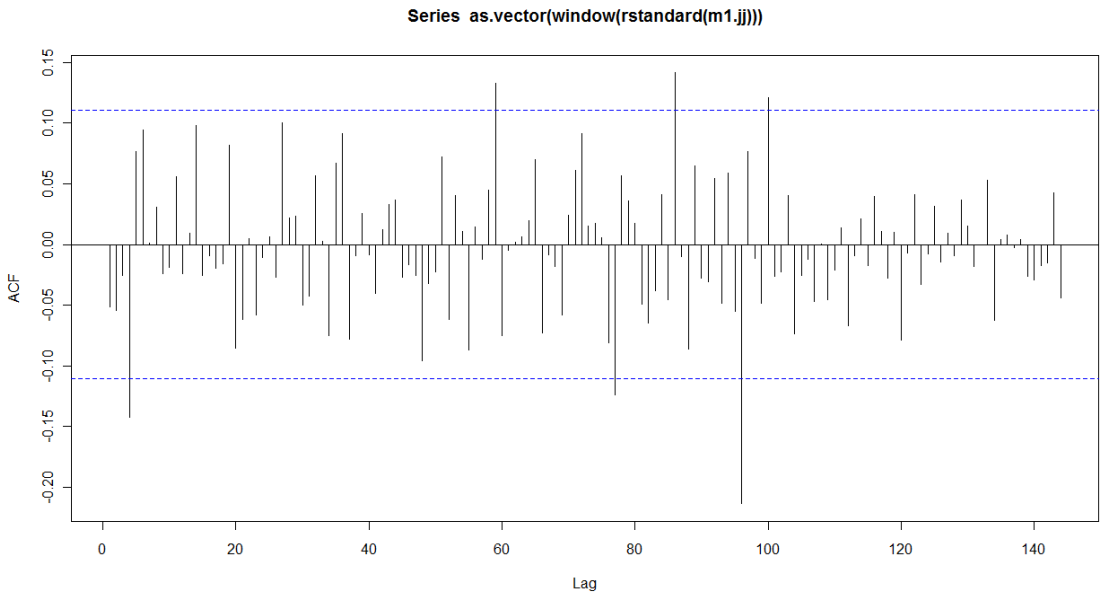
RUTA 1:

```
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)
```



Hay muchos picos que superan el umbral habitual de 2 y -2. Para precisar más, visualizaremos el ACF de los residuales.

```
acf(as.vector(window(rstandard(m1.jj))),lag.max=144)
```



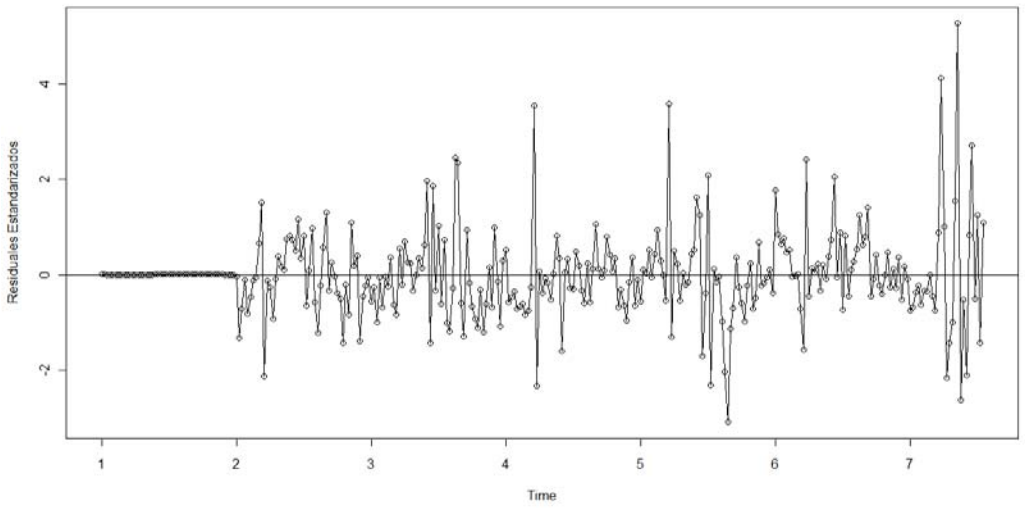
Se aprecian seis correlaciones "estadísticamente significativas". La más significativa, situada en el retraso 96 presenta un valor de correlación que supera el -0,20 (una correlación muy pequeña) y podría ser debido a que es múltiplo de la frecuencia de datos, 48. Así, se puede decir que el modelo ha capturado la esencia de la dependencia de la serie.

Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

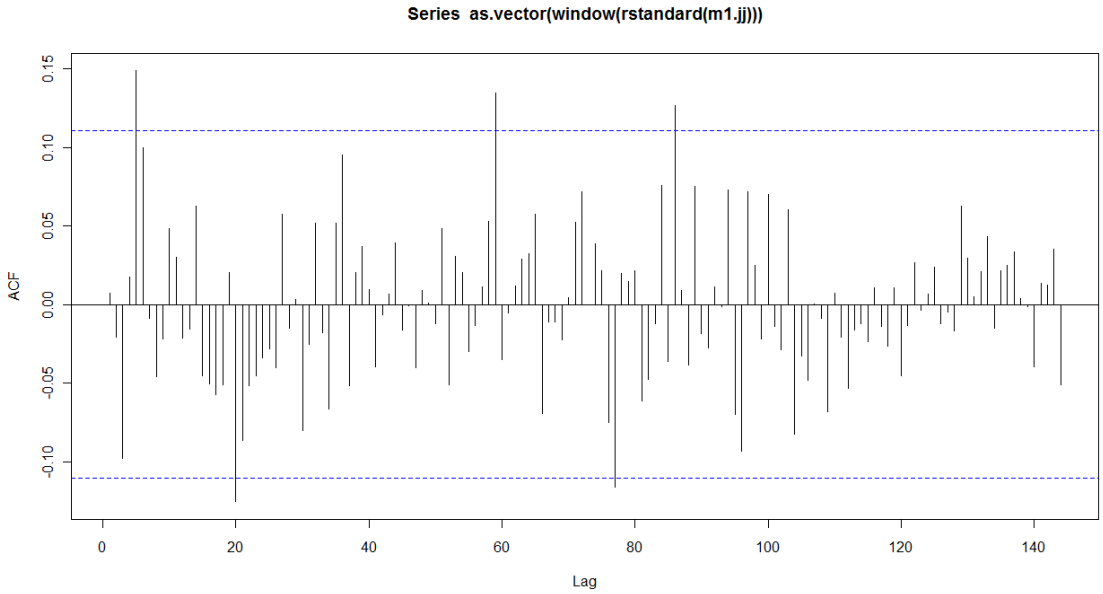
var(jj)
[1] 9.851488e+14
SNR = 9.851488e+14 / 5.671e+13 = 17,37

RUTA 2:

```
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)
```



Los picos que presenta la gráfica son aproximadamente los mismos que para la ruta 1. Para precisar más, visualizaremos el ACF de los residuales.
acf(as.vector(window(rstandard(m1.jj))),lag.max=144)



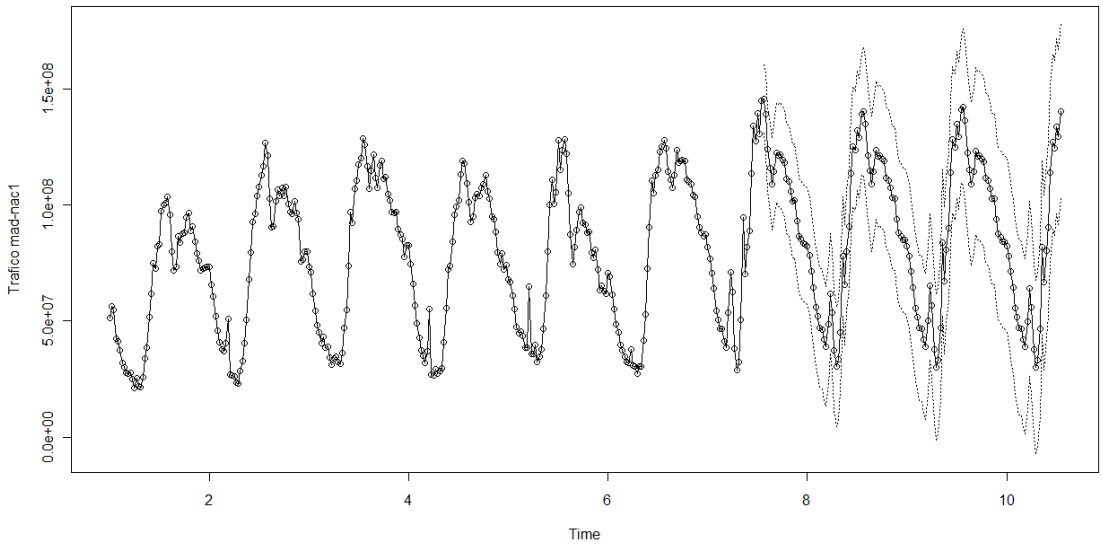
Decididamente, las correlaciones significativas se han visto reducidas en número y en fuerza con una correlación más significativa de valor 0,15. Además, no tenemos una clara razón para pensar que los retrasos se deban a algo más que a la casualidad.

Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

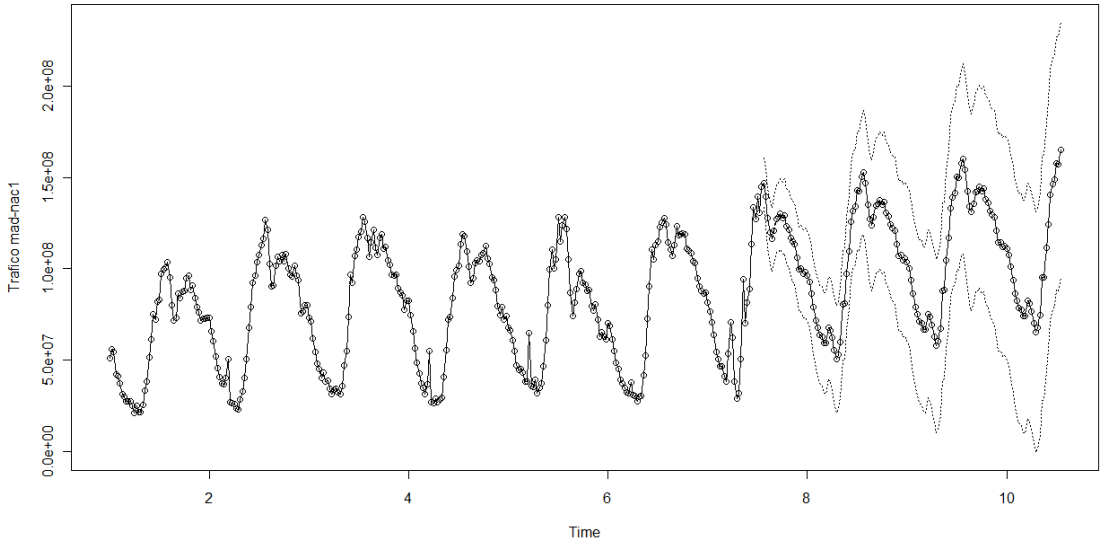
var(jj)
[1] 9.851488e+14
SNR = 9.851488e+14 / 4.693e+13 = 20,99

Si comparamos los resultados de ambas rutas, la ruta 2 presenta una mejor relación señal-ruido y unas correlaciones de los residuales más bajas, lo que parece indicar que el modelo tiene una mejor adecuación a la serie temporal. Esto se corresponde con lo indicado por los atributos AIC retornados para los dos modelos, ya que el menor AIC lo presenta el modelo de la segunda ruta. No obstante, la ruta 2 presenta una mayor varianza en las predicciones de la que presenta la ruta 1.

PREDICCIÓN
plot(m1.jj,n.ahead=144,xlab='Time',type='o',ylab='Trafico mad-nac1')
ARIMA $(4,0,0)x(1,1,0)_{48}$



ARIMA $(0,1,2)x(0,1,1)_{48}$



La versión extendida:

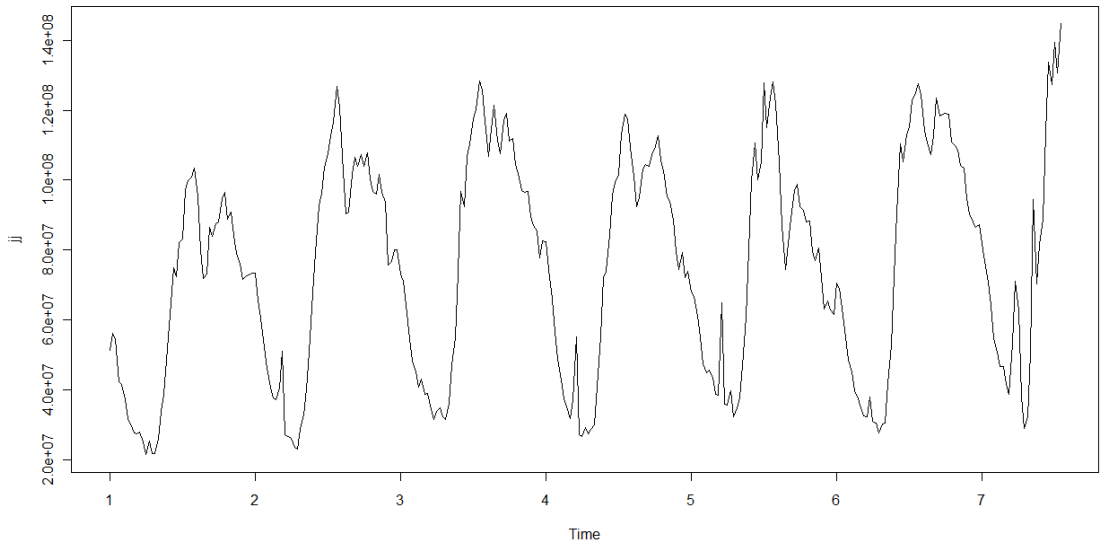
ESPECIFICACIÓN DEL MODELO

- Carga de datos:

jj = ts(scan("c:/Users/Santi/Documents/TFC/Datos DAT/Partidos/mad-nac1-lab.dat"), frequency=48)

- Imagen:

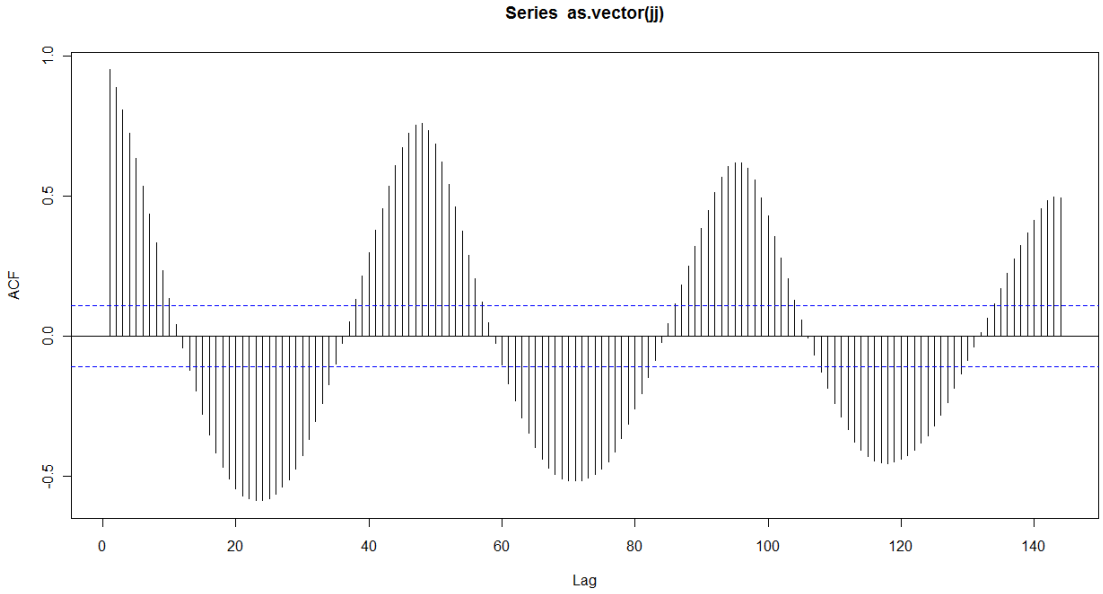
plot(jj)



No se aprecia una tendencia ascendente que haga pensar en una necesaria primera diferenciación (parece ser un modelo estacionario). Sí se observa una clara estacionalidad.

- ACF:

acf(as.vector(jj),lag.max=144)



La estacionalidad queda verificada.

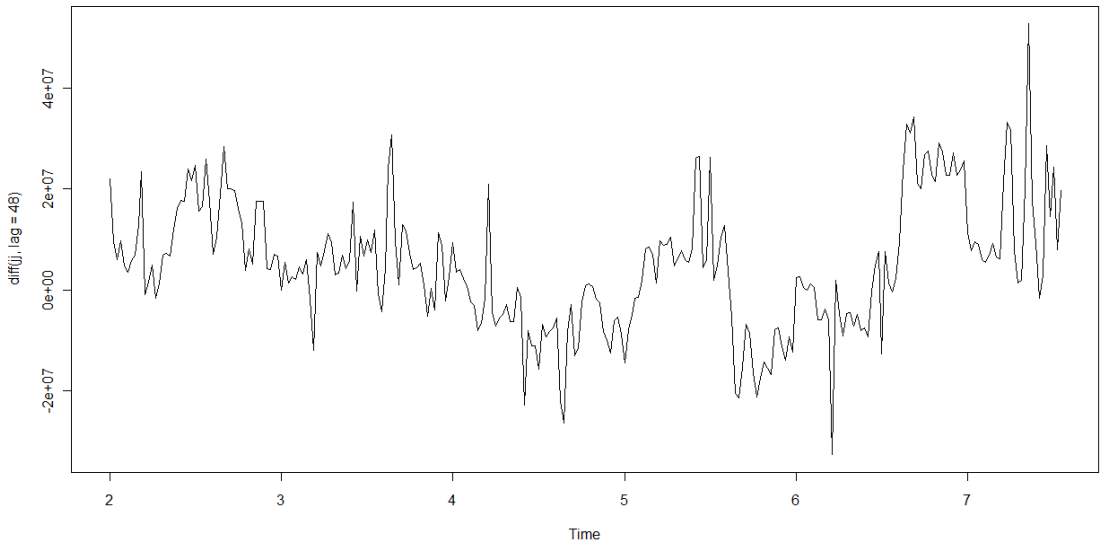
A partir de este punto llevaremos a cabo 2 rutas para la especificación del modelo. La primera en la que sólo derivaremos estacionalmente y la segunda en la que además de derivar estacionalmente, aplicaremos la primera derivada sobre el modelo. Partiendo de la necesidad de eliminar la estacionalidad a 48, ¿porqué aplicar también la primera derivada al modelo? La segunda ruta tiene como objetivo ver si hay una no estacionaridad que no se ha observado una vez aplicada la derivada estacional. Tomando estas 2 rutas más probables, esperamos conseguir 2 modelos bastante adecuados, eligiendo después el

más prometedor (un compromiso entre simplicidad y adecuación). Se aplicará el siguiente criterio: para este caso, se tomarán TODAS las correlaciones que superen el umbral del 95%.

RUTA 1:

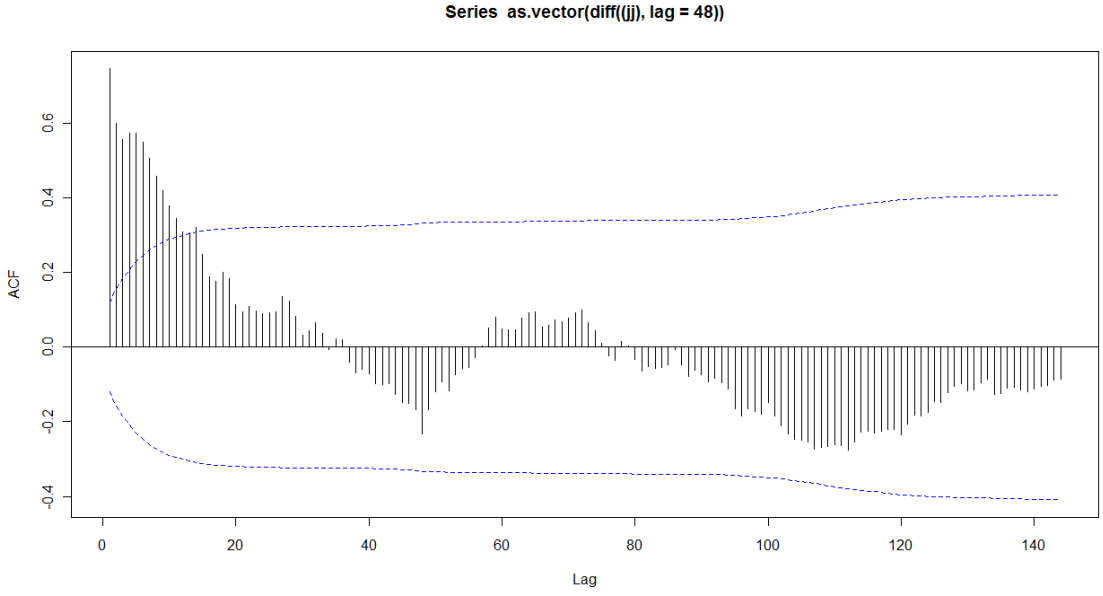
- Derivando estacionalmente con lag = 48 (48 son los datos de un día)

plot(diff(jj,lag=48))



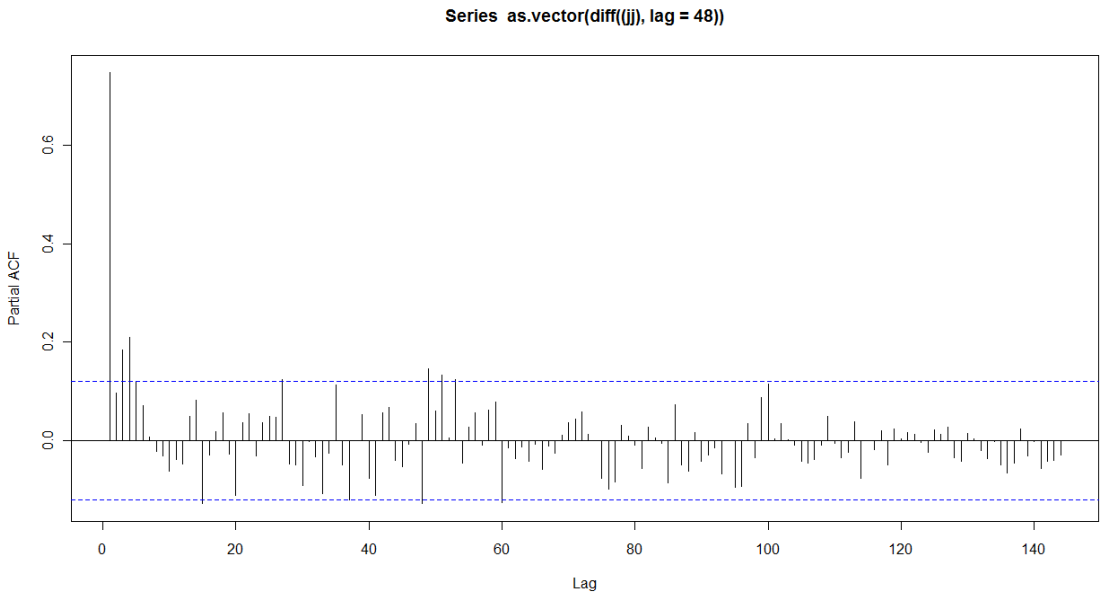
o Aplicando el ACF sobre esta transformación:

acf(as.vector(diff((jj),lag=48)),lag.max=144,ci.type='ma')



o Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff((jj),lag=48)),lag.max=144)

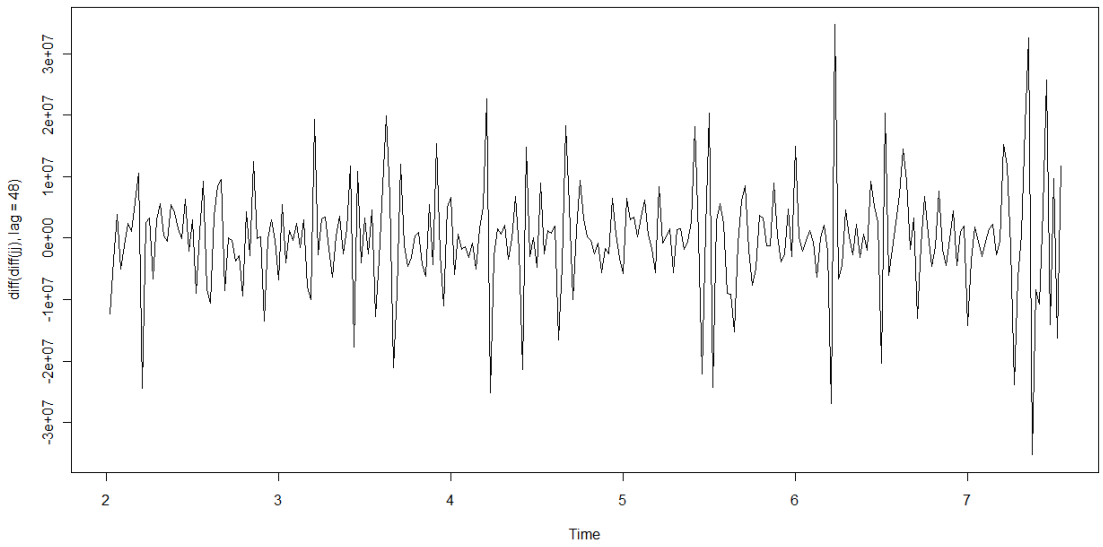


Se puede sugerir que un modelo que incorpore los retrasos 15 y 48 sería adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(15,0,0)x(1,1,0)_{48}$

RUTA 2:

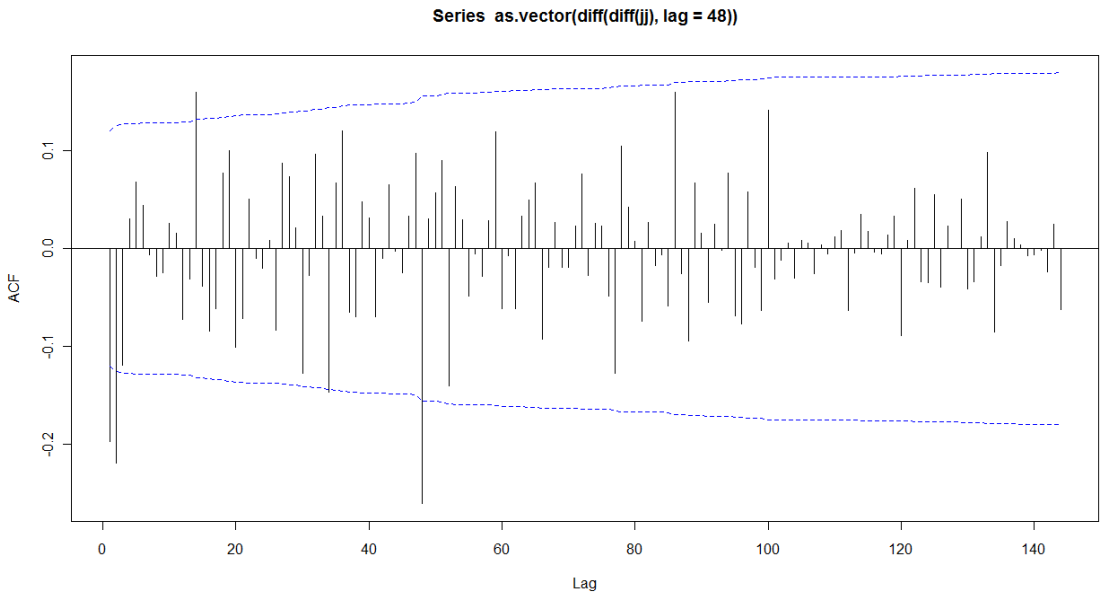
- Derivando estacionalmente con lag = 48 y con la primera derivada

plot(diff(diff(jj),lag=48))



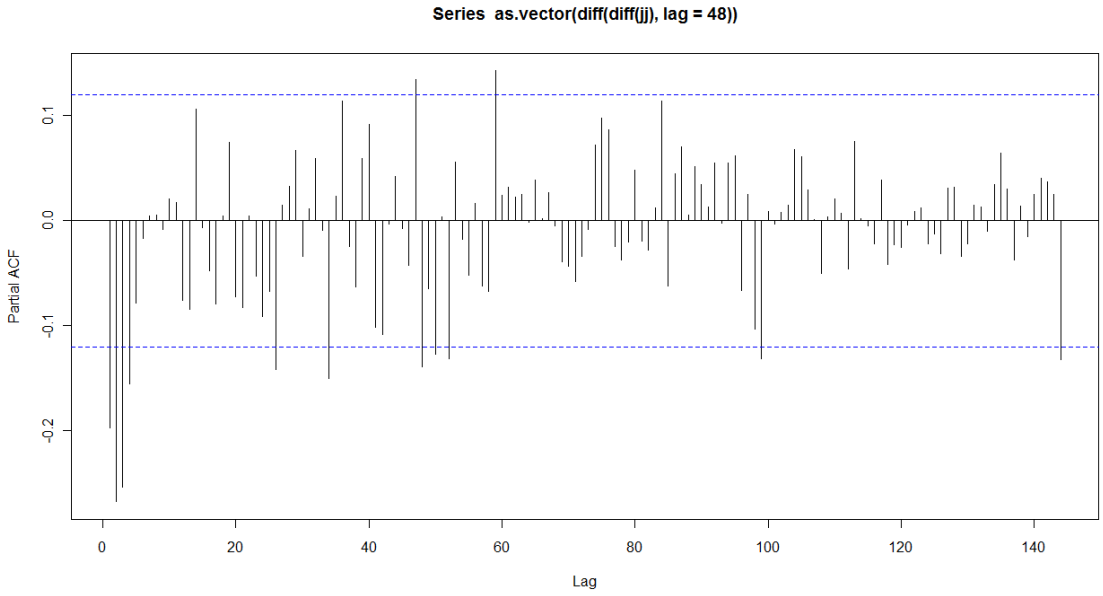
- Aplicando el ACF sobre esta transformación:

acf(as.vector(diff(diff(jj),lag=48)),lag.max=144,ci.type='ma')



- Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff(diff(jj),lag=48)),lag.max=144)



Si se vuelve a observar la gráfica del ACF, se puede sugerir que un modelo que incorpore los retrasos 14 y 48 autocorrelativos seria adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(0,1,14)x(0,1,1)_{48}$

ESTIMACIÓN DE LOS PARÁMETROS

RUTA 1:

```
m1.jj=arima(jj,order=c(15,0,0),seasonal=list(order=c(1,1,0),period=48))
m1.jj
Call:
arima(x = jj, order = c(15, 0, 0), seasonal = list(order = c(1, 1, 0), period = 48))
```

Coefficients:

	ar1	ar2	ar3	ar4	ar5	ar6	ar7	ar8	ar9
	0.5940	-0.0549	0.0679	0.1062	0.1631	0.0553	-0.0169	0.0345	-0.0182
s.e.	0.0615	0.0717	0.0723	0.0725	0.0738	0.0744	0.0750	0.0748	0.0751

	ar10	ar11	ar12	ar13	ar14	ar15	sar1
	-0.0035	0.0273	-0.0651	0.0231	0.0982	-0.1000	-0.3611
s.e.	0.0804	0.0808	0.0799	0.0796	0.0821	0.0693	0.0681

sigma^2 estimated as 5.353e+13: log likelihood = -4603.21, aic = 9238.42

RUTA 2:

```
m1.jj=arima(jj,order=c(0,1,14),seasonal=list(order=c(0,1,1),period=48))
m1.jj
```

Call:
arima(x = jj, order = c(0, 1, 14), seasonal = list(order = c(0, 1, 1), period = 48))

Coefficients:

	ma1	ma2	ma3	ma4	ma5	ma6	ma7	ma8
	-0.4046	-0.2805	-0.0944	0.0254	0.1470	0.0855	-0.0626	-0.1458
s.e.	0.0617	0.0678	0.0704	0.0755	0.0734	0.0947	0.0750	0.1099

	ma9	ma10	ma11	ma12	ma13	ma14	sma1
	-0.0437	0.1495	0.1038	-0.0981	-0.1310	0.0704	-0.7671
s.e.	0.0896	0.1111	0.1049	0.1009	0.1076	0.1091	0.1216

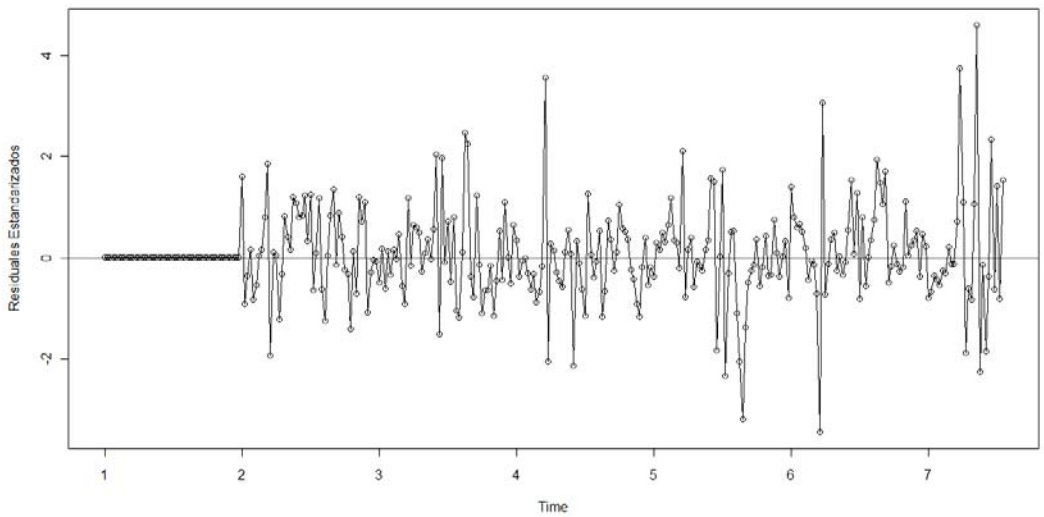
sigma^2 estimated as 4.292e+13: log likelihood = -4573.46, aic = 9176.91

CHEQUEO DEL DIAGNÓSTICO

A continuación procederemos a comprobar la bondad del afinado de los modelos y para ello, primero hay que observar los residuales estandarizados.

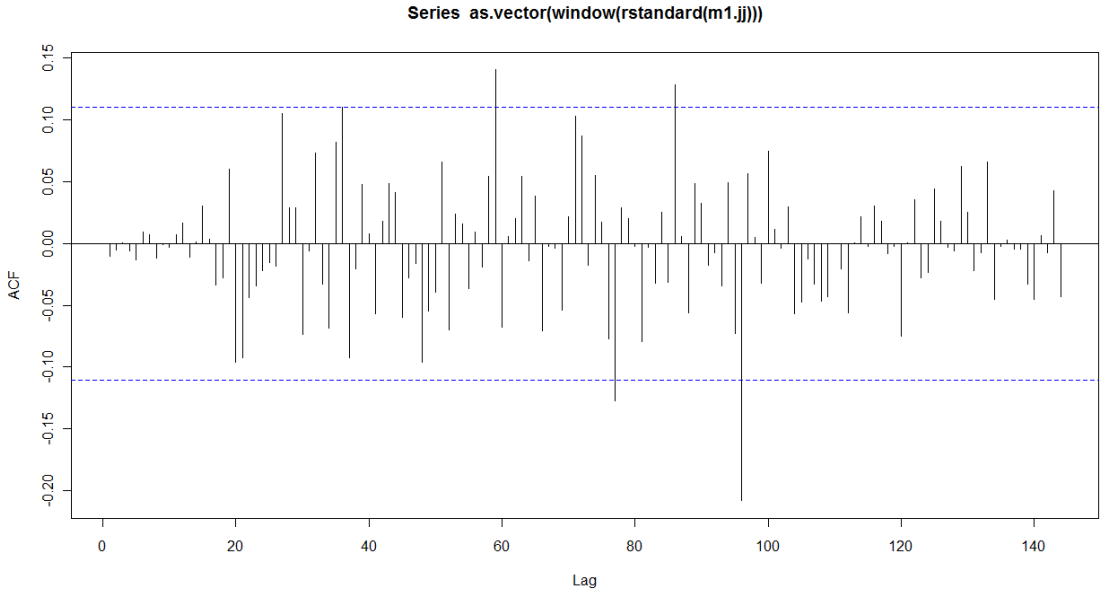
RUTA 1:

```
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)
```



Hay muchos picos que superan el umbral habitual de 2 y -2. Para precisar más, visualizaremos el ACF de los residuales.

acf(as.vector(window(rstandard(m1.jj))),lag.max=144)

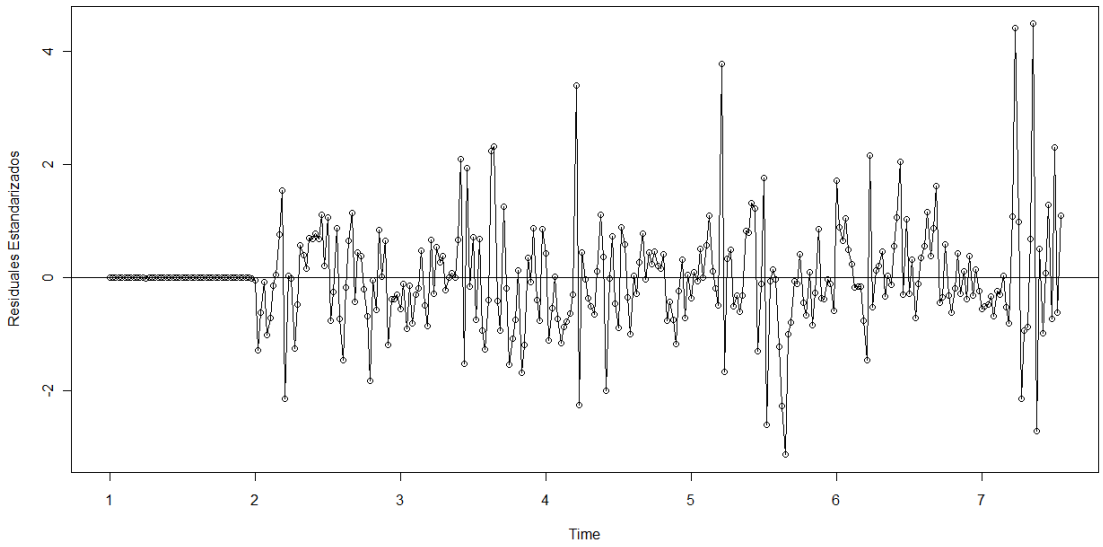


Se aprecian cuatro correlaciones “estadísticamente significativas”, frente a las seis del modelo reducido. La más significativa está situada en el retraso 96 y sigue presentando un valor de correlación que supera el -0,20.

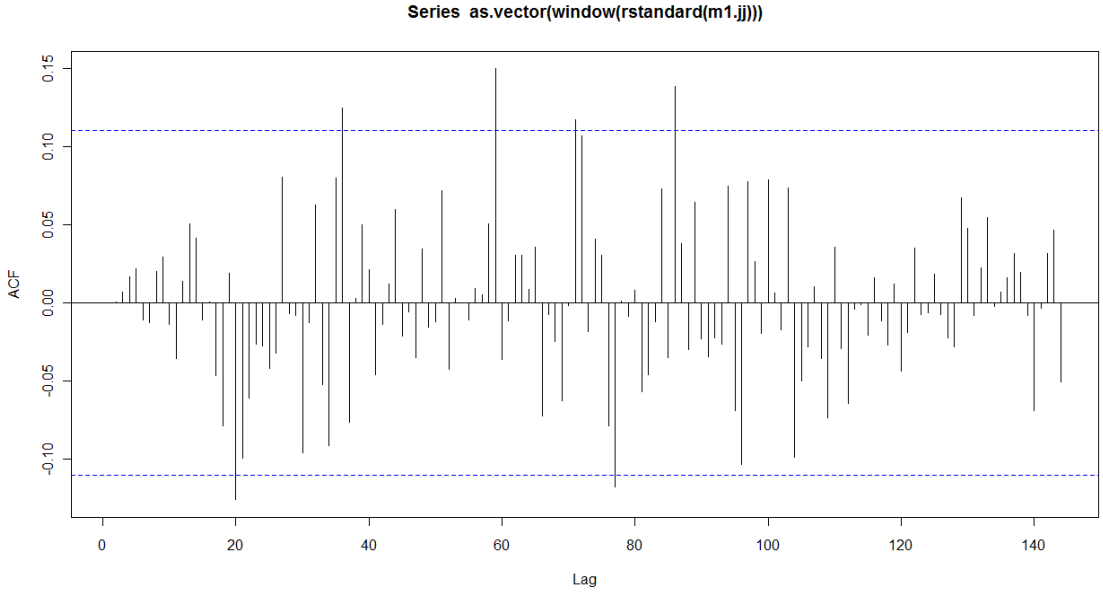
Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

var(jj)
[1] 9.851488e+14
SNR = 9.851488e+14 / 5.353e+13 = 18,40
El SNR ha mejorado respecto el modelo reducido, pero el AIC ha empeorado.

RUTA 2:
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)



Los picos que presenta la gráfica son aproximadamente los mismos que para la ruta 1. Para precisar más, visualizaremos el ACF de los residuales.
acf(as.vector(window(rstandard(m1.jj))),lag.max=144)



En este caso, también hay 6 correlaciones significativas, con la más fuerte de valor 0,15.

Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

var(jj)
[1] 9.851488e+14
SNR = 9.851488e+14 / 4.292e+13 = 22,95

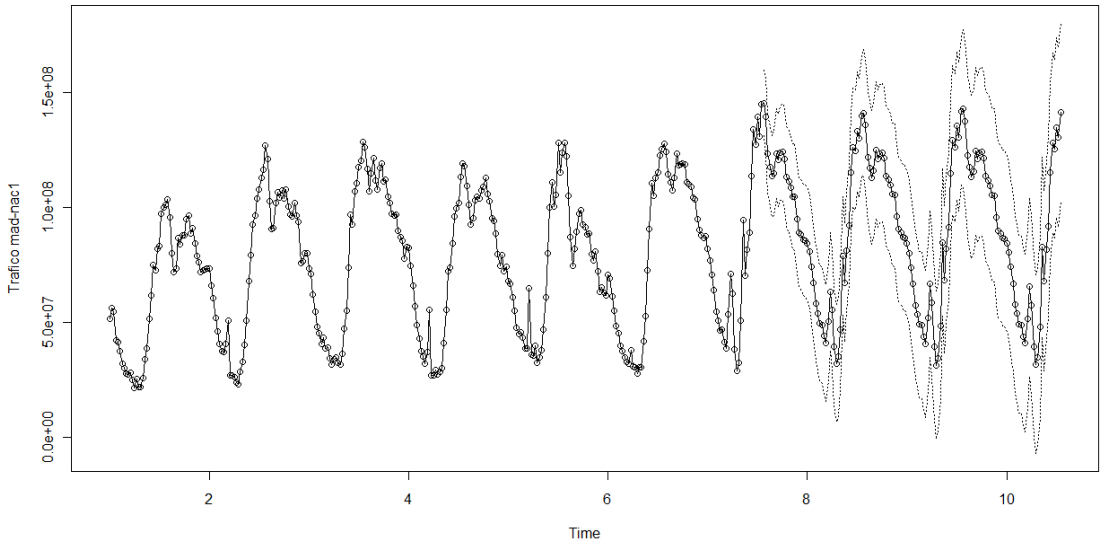
En este caso, la sobreadecuación del modelo ha conducido a la aparición de más correlaciones y a un empeoramiento del valor del AIC, sin embargo, la SNR ha mejorado respecto al modelo reducido.

Si comparamos los resultados de ambas rutas, la ruta 2 presenta una mejor relación señal-ruido y unas correlaciones de los residuales más bajas, lo que parece indicar que el modelo tiene una mejor adecuación a la serie temporal. Esto se corresponde con lo indicado por los atributos AIC retornados para los dos modelos, ya que el menor AIC lo presenta el modelo de la segunda ruta. No obstante, la ruta 2 presenta una mayor varianza en las predicciones de la que presenta la ruta 1.

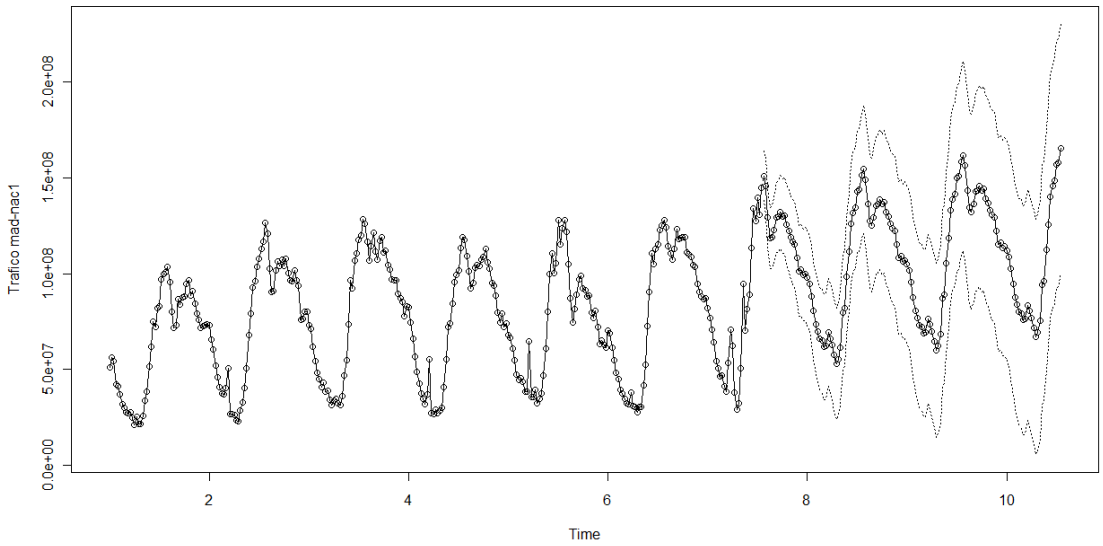
PREDICCIÓN

plot(m1.jj,n.ahead=144,xlab='Time',type='o',ylab='Trafico mad-nac1')

ARIMA $(15,0,0)x(1,1,0)_{48}$



ARIMA $(0,1,14)x(0,1,1)_{48}$



A.III.4. Enlace nac-and

La versión limitada:

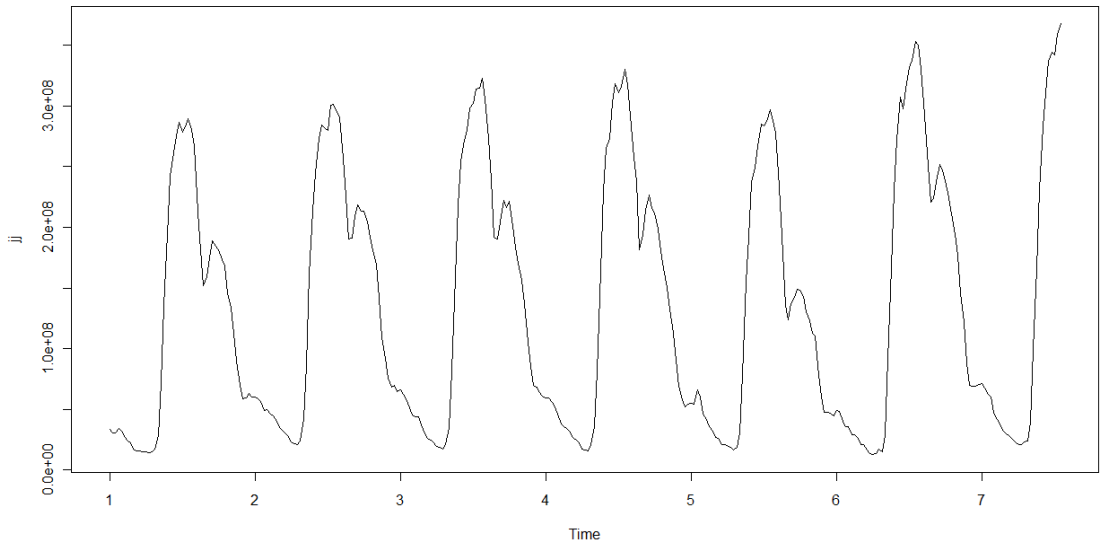
ESPECIFICACIÓN DEL MODELO

- Carga de datos:

jj = ts(scan("c:/Users/Santi/Documents/TFC/Datos DAT/Partidos/nac-and-lab.dat"), frequency=48)

- Imagen:

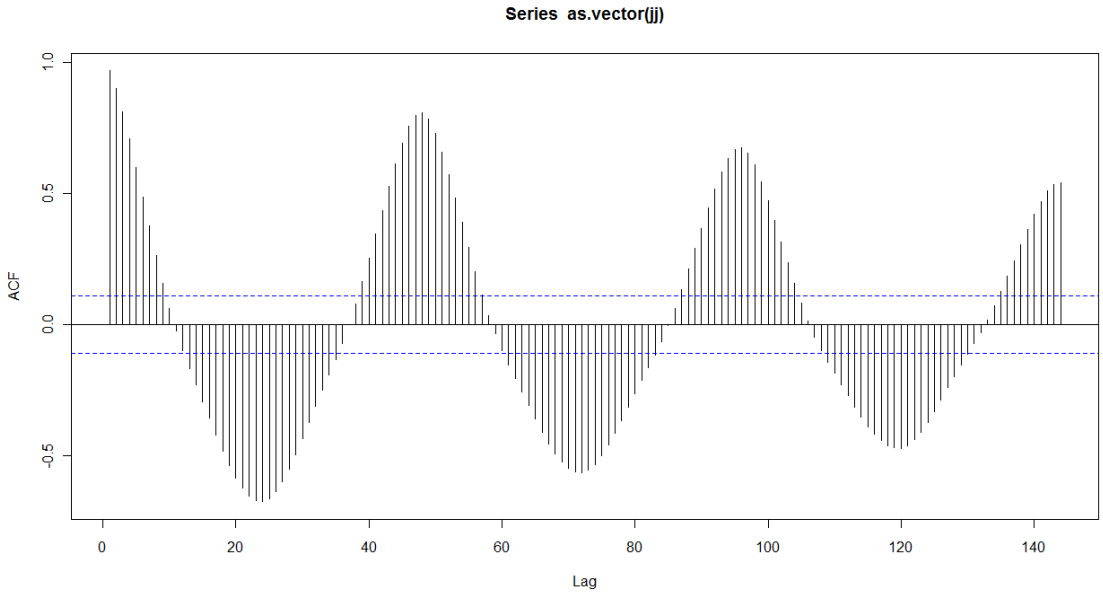
plot(jj)



No se aprecia una tendencia ascendente que haga pensar en una necesaria primera diferenciación (parece ser un modelo estacionario). Sí se observa una clara estacionalidad.

- ACF:

acf(as.vector(jj),lag.max=144)



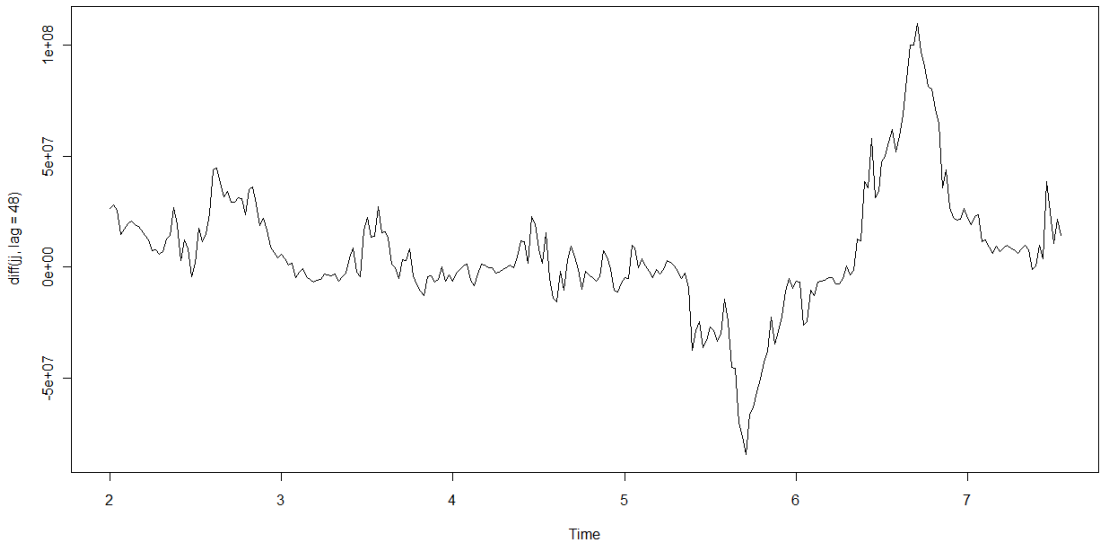
La estacionalidad queda verificada.

A partir de este punto llevaremos a cabo 2 rutas para la especificación del modelo. La primera en la que sólo derivaremos estacionalmente y la segunda en la que además de derivar estacionalmente, aplicaremos la primera derivada sobre el modelo. Partiendo de la necesidad de eliminar la estacionalidad a 48, ¿porqué aplicar también la primera derivada al modelo? La segunda ruta tiene como objetivo ver si hay una no estacionaridad que no se ha observado una vez aplicada la derivada estacional. Tomando estas 2 rutas más probables, esperamos conseguir 2 modelos bastante adecuados, eligiendo después el más prometedor (un compromiso entre simplicidad y adecuación). Se aplicará el siguiente criterio: las correlaciones que se considerarán significativas serán aquellas que superen un factor de 0,2 y con no más de 4 retrasos que superen el umbral del 95%, admitiendo el retraso 48 como necesario a nivel estacional.

RUTA 1:

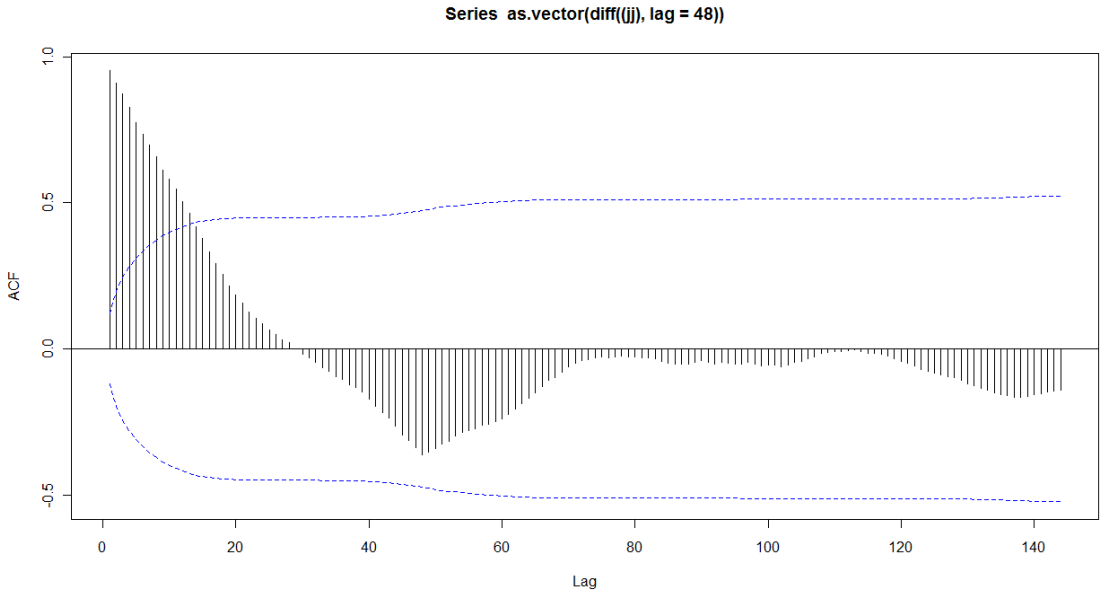
- Derivando estacionalmente con lag = 48 (48 son los datos de un día)

```
plot(diff(jj,lag=48))
```



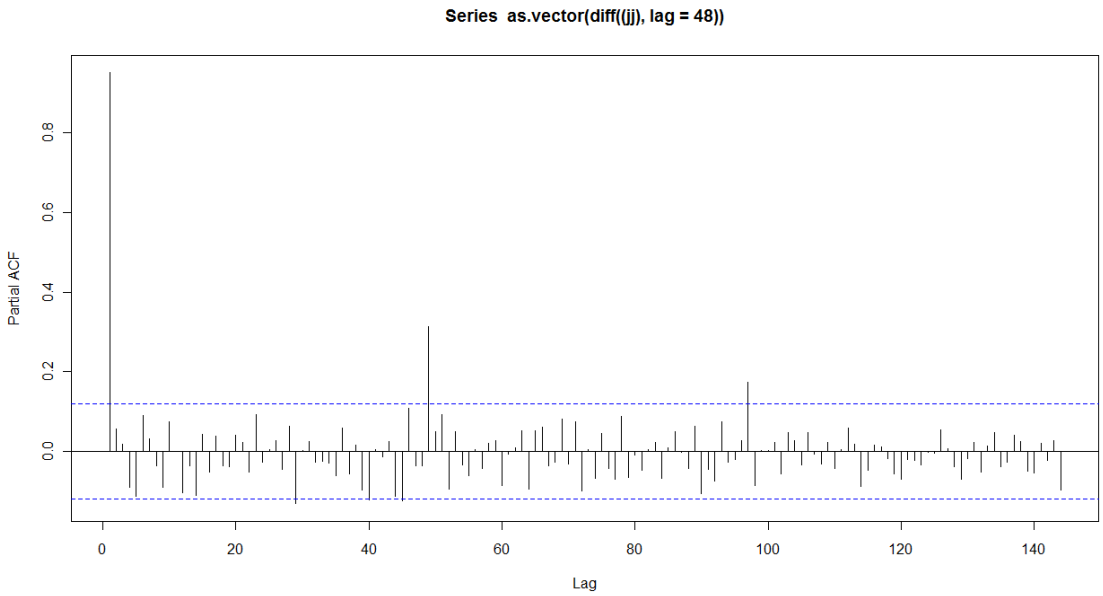
- Aplicando el ACF sobre esta transformación:

acf(as.vector(diff((jj),lag=48)),lag.max=144,ci.type='ma')



- Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff((jj),lag=48)),lag.max=144)

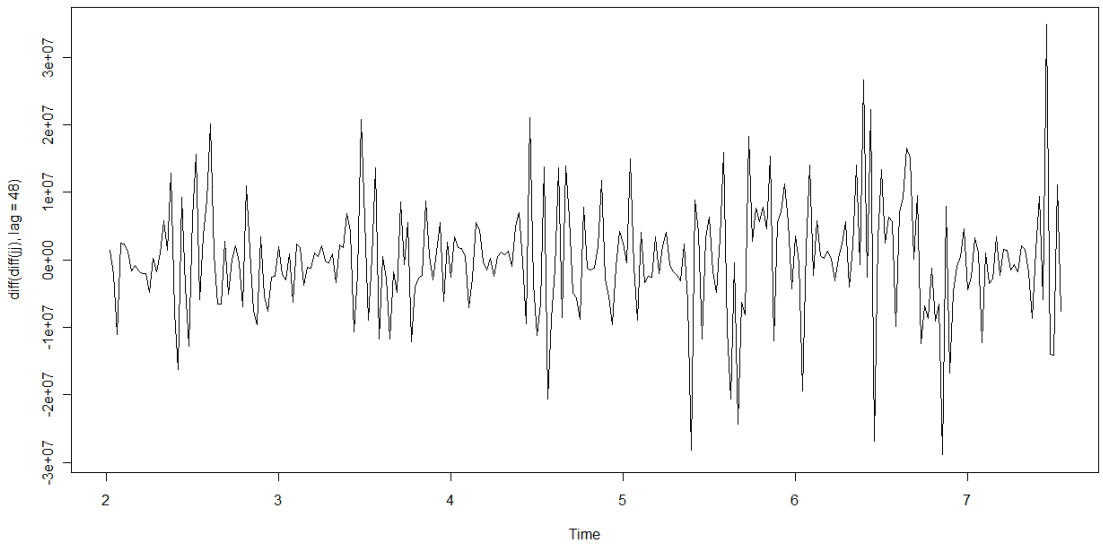


Se puede sugerir que un modelo que incorpore los retrasos 1 y 48 sería adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(1,0,0)x(1,1,0)_{48}$

RUTA 2:

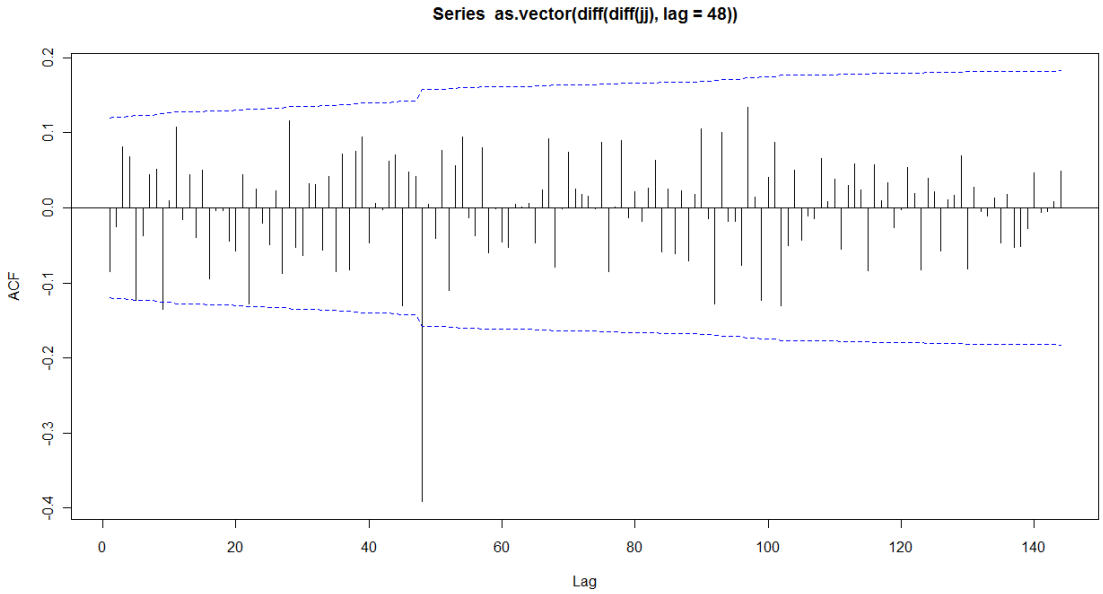
- Derivando estacionalmente con lag = 48 y con la primera derivada

plot(diff(diff(jj),lag=48))



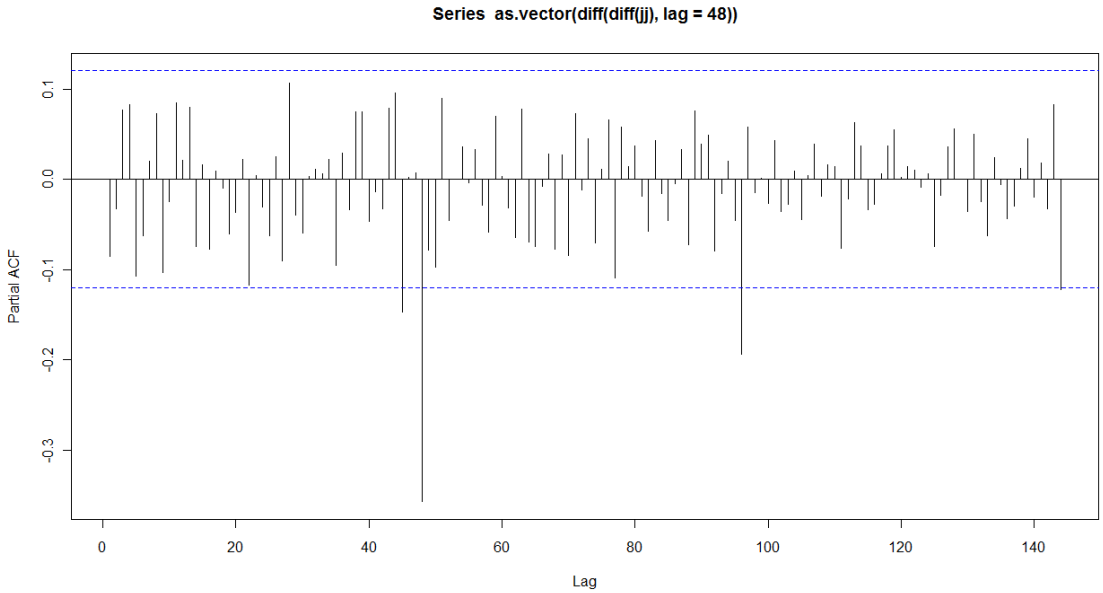
- Aplicando el ACF sobre esta transformación:

acf(as.vector(diff(diff(jj),lag=48)),lag.max=144,ci.type='ma')



- Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff(diff(jj),lag=48)),lag.max=144)



Si se vuelve a observar la gráfica del ACF, se puede sugerir que un modelo que incorpore los retrasos 0 y 48 autocorrelativos seria adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(0,1,0)x(0,1,1)_{48}$

ESTIMACIÓN DE LOS PARÁMETROS

RUTA 1:

```
m1.jj=arima(jj,order=c(1,0,0),seasonal=list(order=c(1,1,0),period=48))
m1.jj
Call:
arima(x = jj, order = c(1, 0, 0), seasonal = list(order = c(1, 1, 0), period = 48))

Coefficients:
```

```
        ar1     sar1
      0.9582  -0.4833
s.e.  0.0182   0.0568

sigma^2 estimated as 5.287e+13:  log likelihood = -4604.87,  aic = 9213.75
```

RUTA 2:

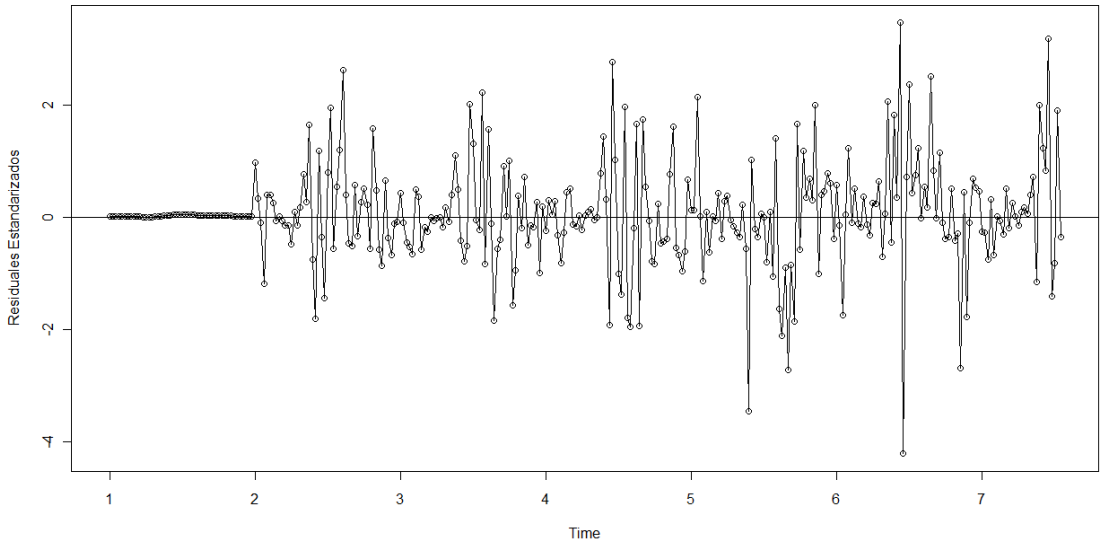
```
m1.jj=arima(jj,order=c(0,1,0),seasonal=list(order=c(0,1,1),period=48))
m1.jj
Call:
arima(x = jj, order = c(0, 1, 0), seasonal = list(order = c(0, 1, 1), period = 48))

Coefficients:
         sma1
      -0.9998
s.e.   0.1607

sigma^2 estimated as 3.403e+13:  log likelihood = -4566.46,  aic = 9134.92
```

CHEQUEO DEL DIAGNÓSTICO

A continuación procederemos a comprobar la bondad del afinado de los modelos y para ello, primero hay que observar los residuales estandarizados.

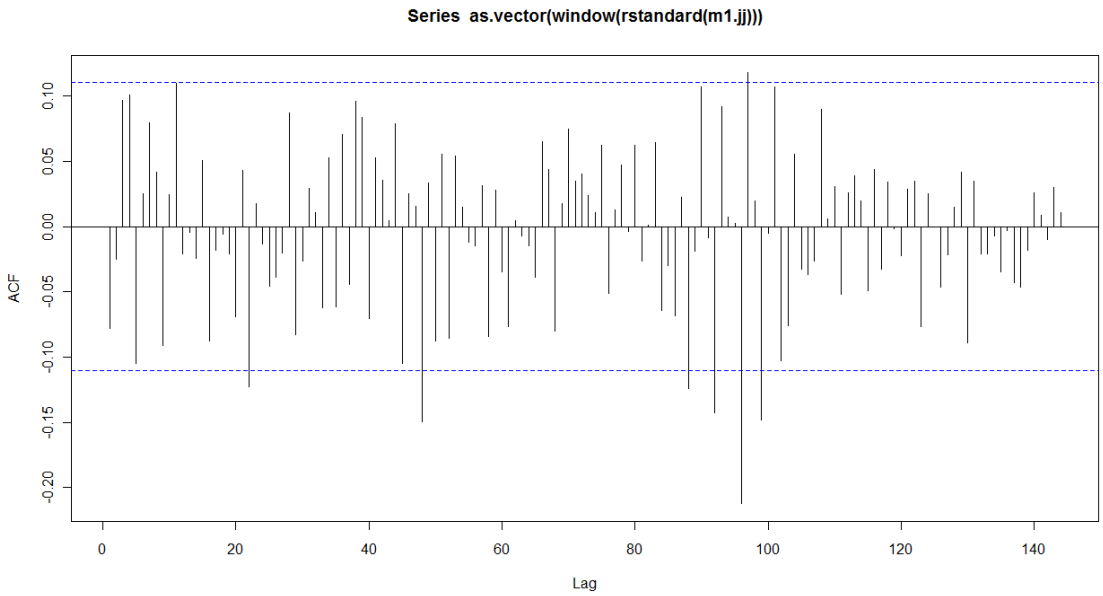
RUTA 1:

```
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)
```



Salvo por una serie de picos a finales de la gráfica, el resto de residuales se encuentran acotados entre 2 y -2 aproximadamente. Para precisar más, visualizaremos el ACF de los residuales.

```
acf(as.vector(window(rstandard(m1.jj))),lag.max=144)
```



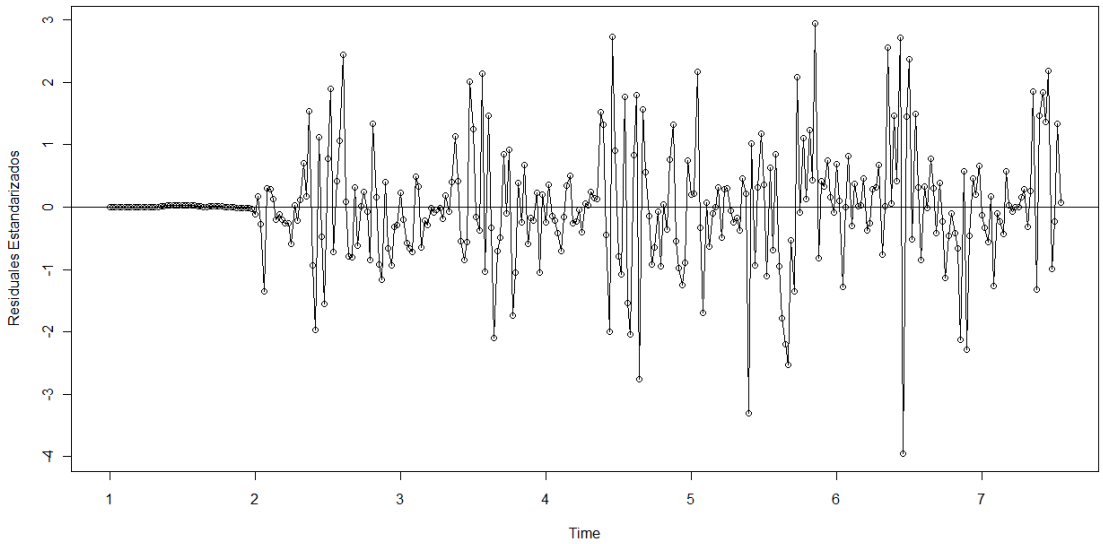
Se aprecian siete correlaciones “estadísticamente significativas”. La correlación más significativa presenta un valor de correlación del entorno del -0,20, un valor no muy elevado.

Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

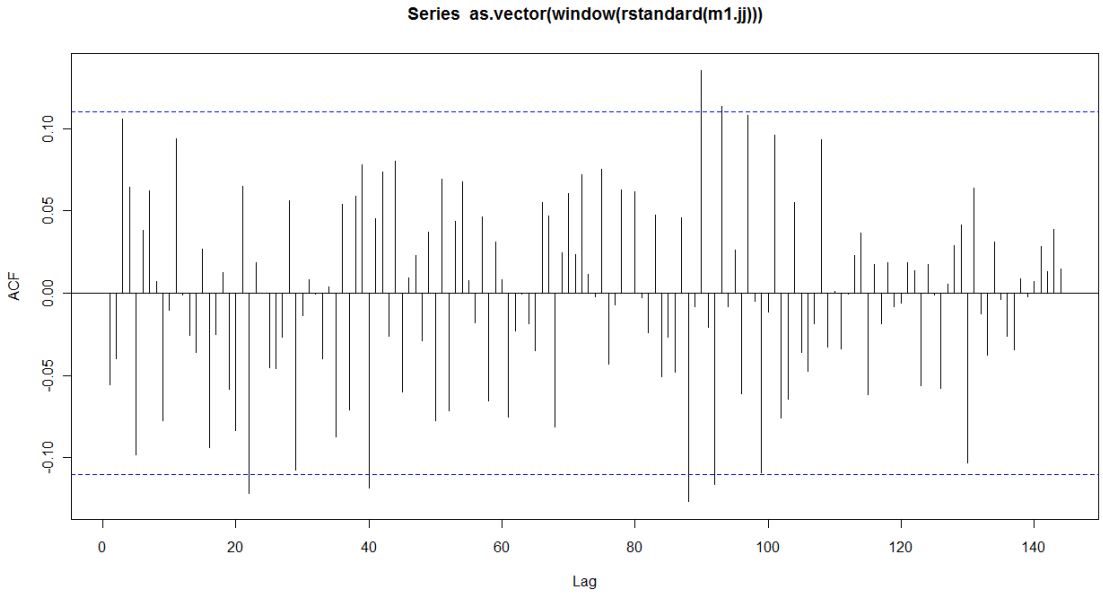
var(jj)
[1] 1.113918e+16
SNR = 1.113918e+16 / 5.287e+13 = 210,69

RUTA 2:

```
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)
```



Los picos que presenta la gráfica son aproximadamente los mismos que para la ruta 1. Para precisar más, visualizaremos el ACF de los residuales.
acf(as.vector(window(rstandard(m1.jj))),lag.max=144)



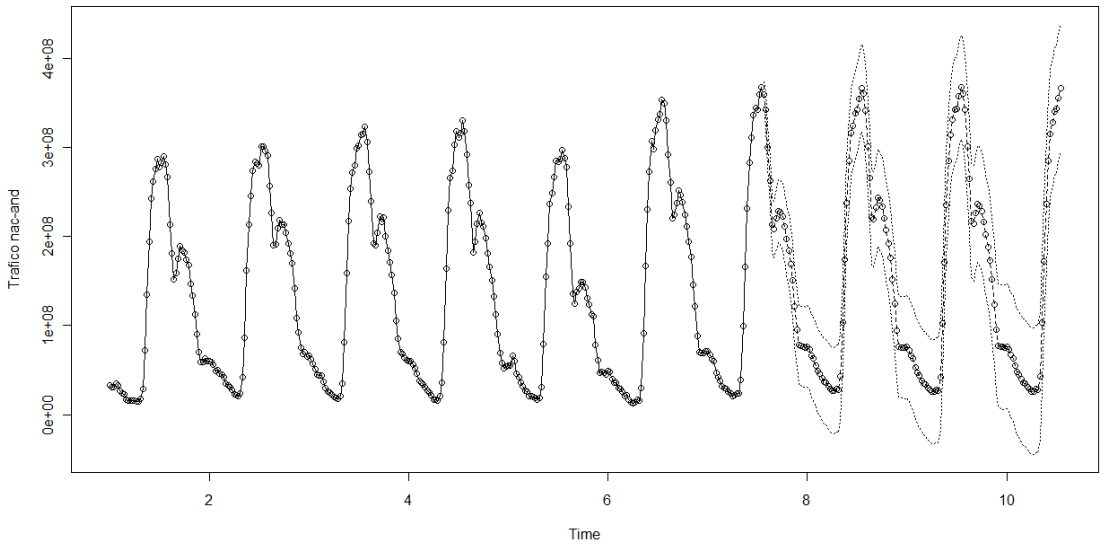
Las seis correlaciones tienen un valor muy bajo.

Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

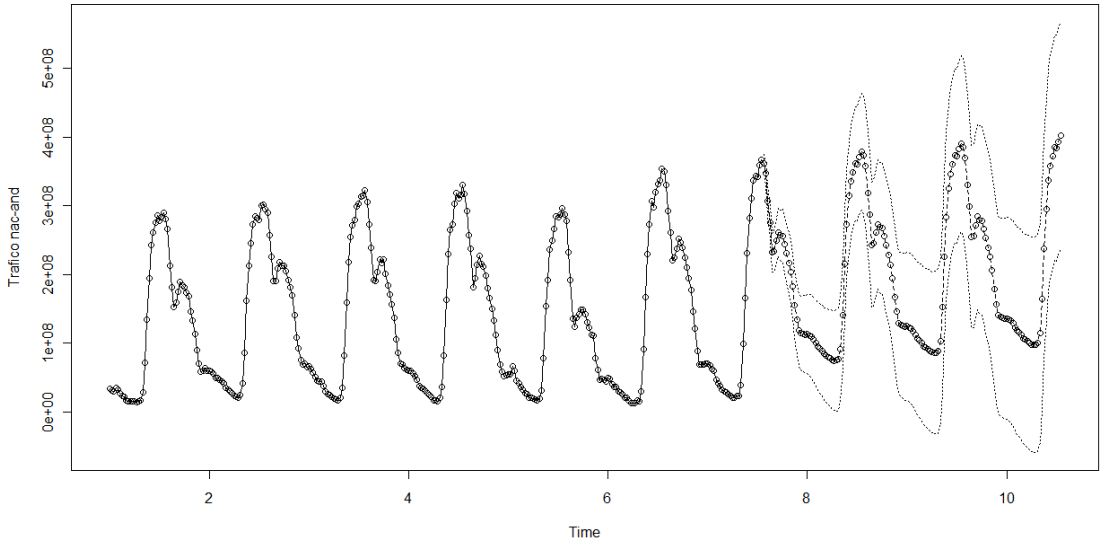
var(jj)
[1] 1.113918e+16
SNR = 1.113918e+16 / 3.403e+13 = 327,33

Si comparamos los resultados de ambas rutas, la ruta 2 presenta una mejor relación señal-ruido y unas correlaciones de los residuales más bajas, lo que parece indicar que el modelo tiene una mejor adecuación a la serie temporal. Esto se corresponde con lo indicado por los atributos AIC retornados para los dos modelos, ya que el menor AIC lo presenta el modelo de la segunda ruta. No obstante, los residuales de la ruta 2 tienen una mayor varianza que los que presenta la ruta 1.

PREDICCIÓN
plot(m1.jj,n.ahead=144,xlab='Time',type='o',ylab='Trafico nac-and')
ARIMA $(1,0,0)x(1,1,0)_{48}$



ARIMA $(0,1,0)x(0,1,1)_{48}$



La versión extendida:

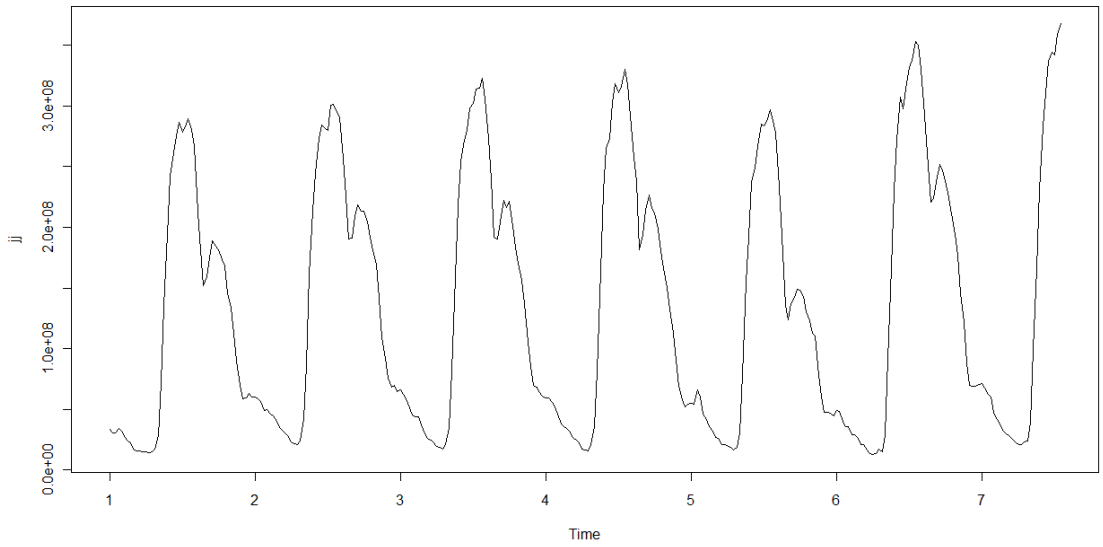
ESPECIFICACIÓN DEL MODELO

- Carga de datos:

jj = ts(scan("c:/Users/Santi/Documents/TFC/Datos DAT/Partidos/nac-and-lab.dat"), frequency=48)

- Imagen:

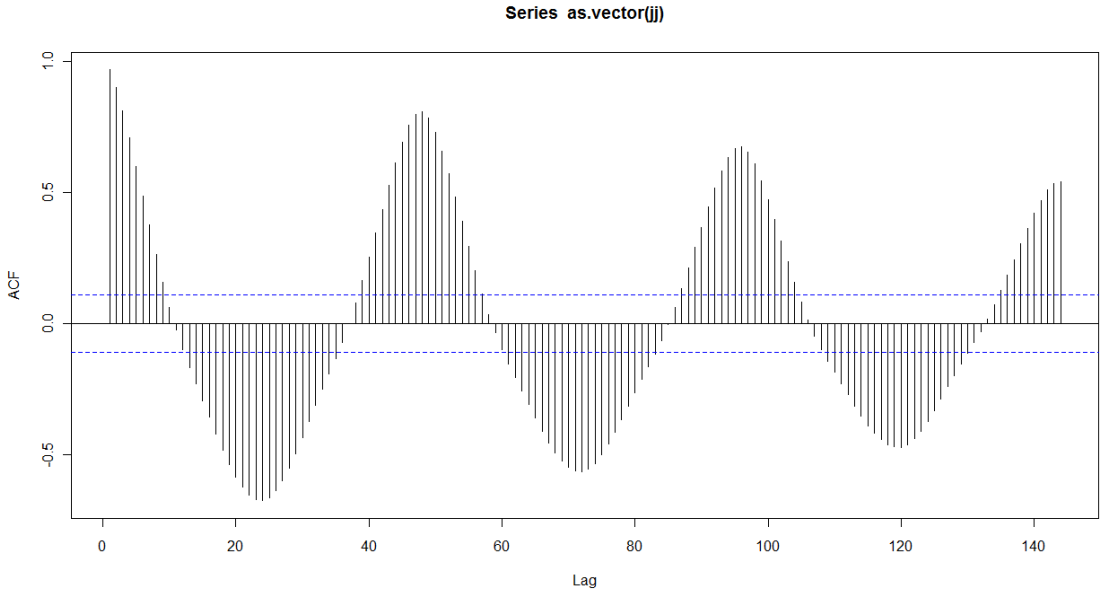
plot(jj)



No se aprecia una tendencia ascendente que haga pensar en una necesaria primera diferenciación (parece ser un modelo estacionario). Sí se observa una clara estacionalidad.

- ACF:

acf(as.vector(jj),lag.max=144)



La estacionalidad queda verificada.

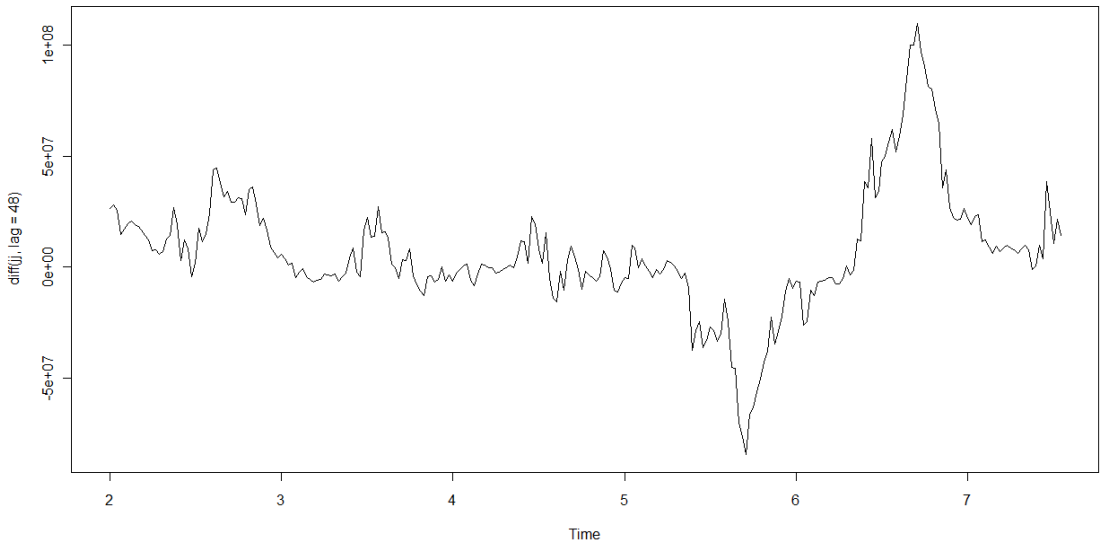
A partir de este punto llevaremos a cabo 2 rutas para la especificación del modelo. La primera en la que sólo derivaremos estacionalmente y la segunda en la que además de derivar estacionalmente, aplicaremos la primera derivada sobre el modelo. Partiendo de la necesidad de eliminar la estacionalidad a 48, ¿porqué aplicar también la primera derivada al modelo? La segunda ruta tiene como objetivo ver si hay una no estacionaridad que no se ha observado una vez aplicada la derivada estacional. Tomando estas 2 rutas más probables, esperamos conseguir 2 modelos bastante adecuados, eligiendo después el

más prometedor (un compromiso entre simplicidad y adecuación). Se aplicará el siguiente criterio: para este caso, se tomarán TODAS las correlaciones que superen el umbral del 95%.

RUTA 1:

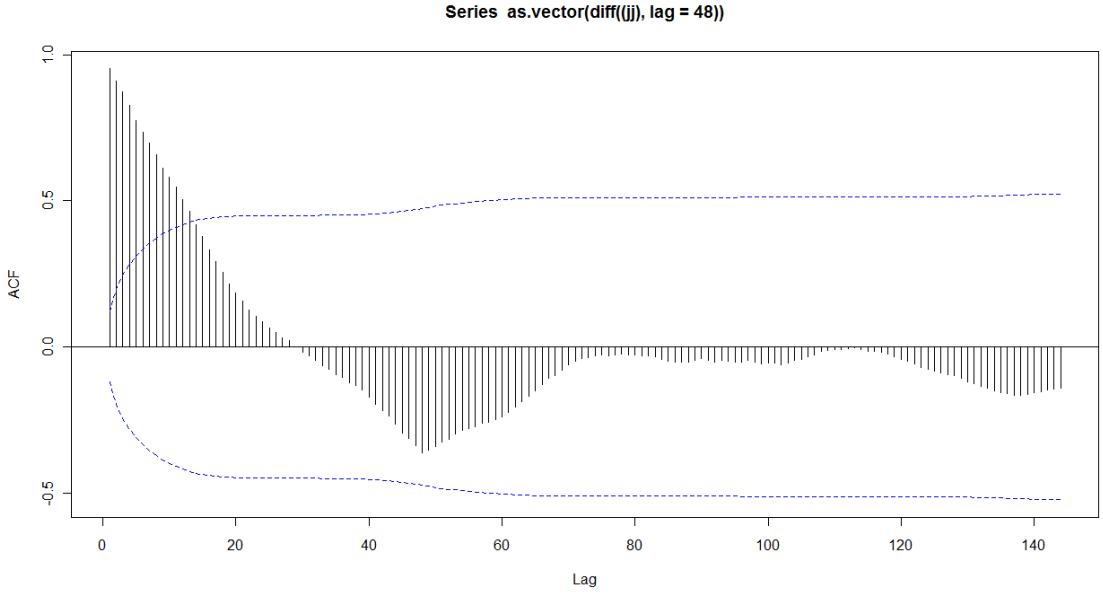
- Derivando estacionalmente con lag = 48 (48 son los datos de un día)

plot(diff(jj,lag=48))



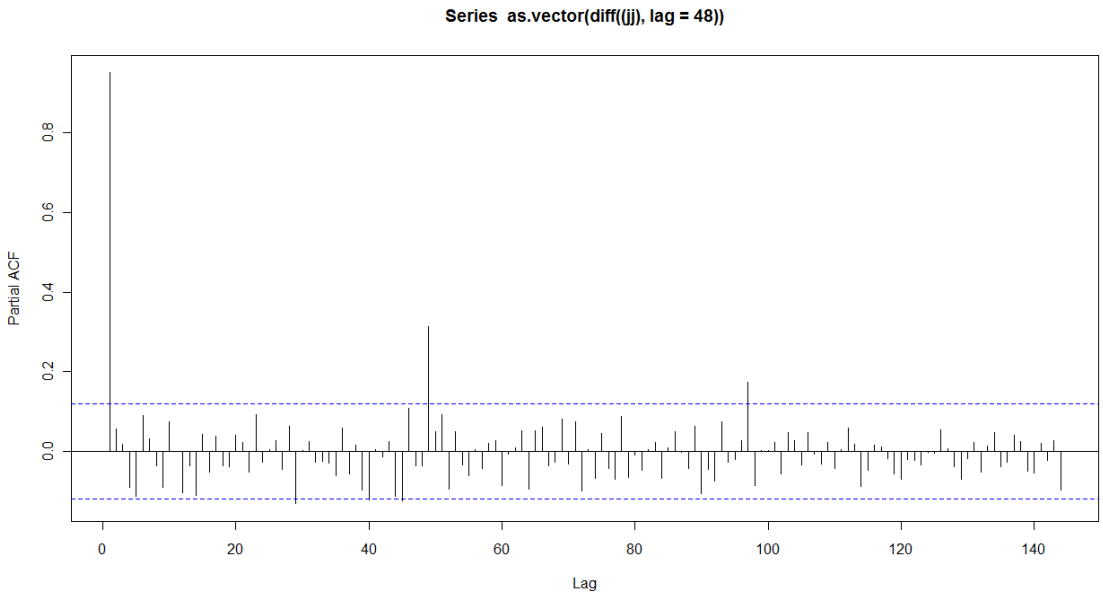
o Aplicando el ACF sobre esta transformación:

acf(as.vector(diff((jj),lag=48)),lag.max=144,ci.type='ma')



o Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff((jj),lag=48)),lag.max=144)



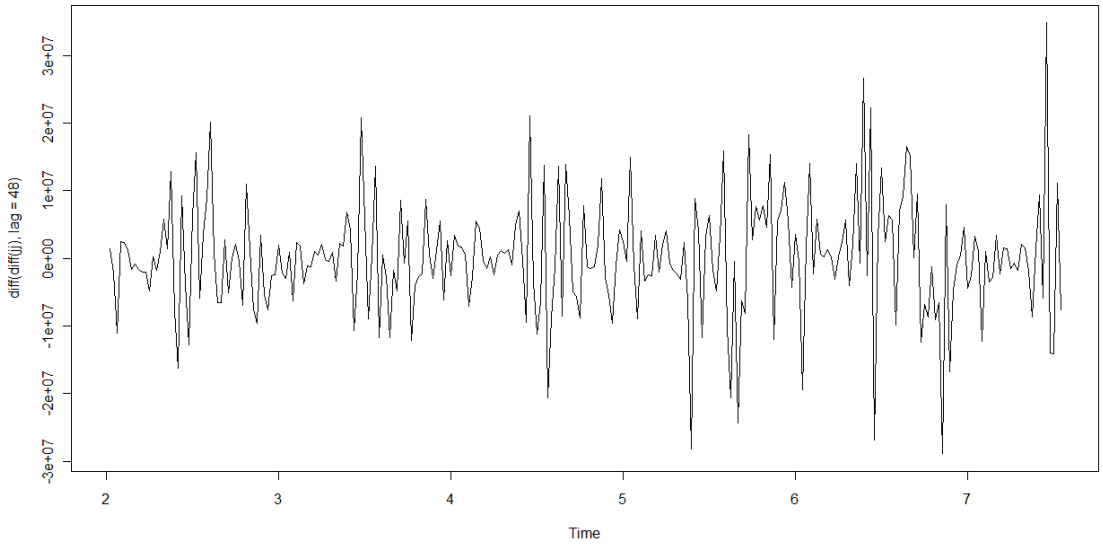
Se puede sugerir que un modelo que incorpore los retrasos 1 y 48 sería adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(1,0,0)x(1,1,0)_{48}$

NOTA: en este caso, el modelo completo coincidirá con el modelo reducido debido a que es el mejor y a la vez único modelo que se puede proponer.

RUTA 2:

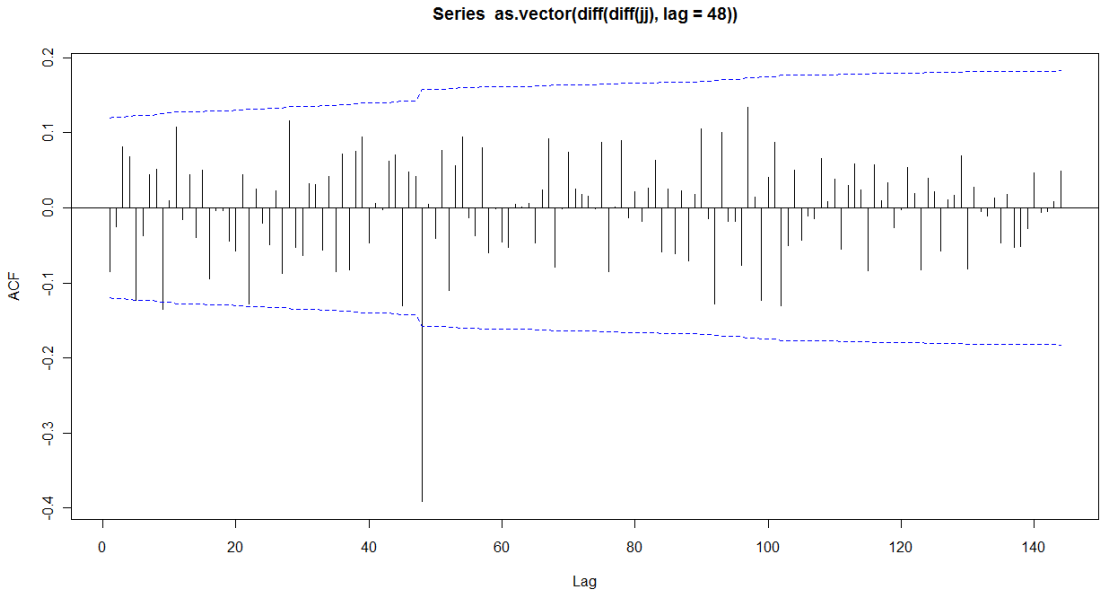
- Derivando estacionalmente con lag = 48 y con la primera derivada

```
plot(diff(diff(jj),lag=48))
```



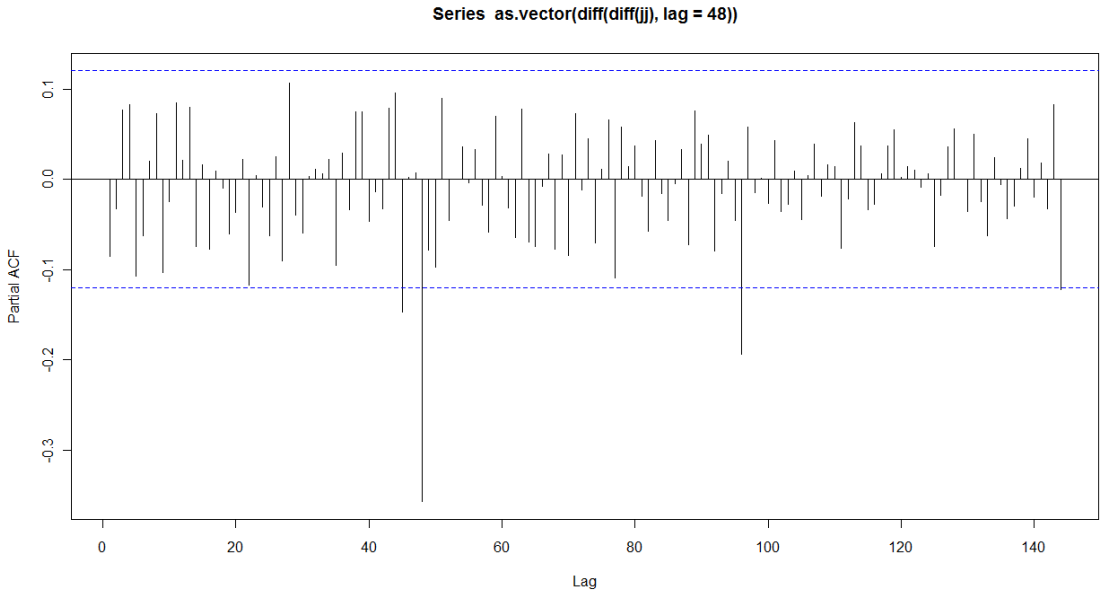
- Aplicando el ACF sobre esta transformación:

```
acf(as.vector(diff(diff(jj),lag=48)),lag.max=144,ci.type='ma')
```



- Aplicando el PACF sobre esta transformación:

pacf(as.vector(diff(diff(jj),lag=48)),lag.max=144)



Si se vuelve a observar la gráfica del ACF, se puede sugerir que un modelo que incorpore los retrasos 9 y 48 autocorrelativos seria adecuado. Para este caso, se trataría de un modelo multiplicativo, estacional ARIMA $(0,1,9)x(0,1,1)_{48}$

ESTIMACIÓN DE LOS PARÁMETROS

RUTA 1:

```
m1.jj=arima(jj,order=c(1,0,0),seasonal=list(order=c(1,1,0),period=48))
m1.jj
Call:
arima(x = jj, order = c(1, 0, 0), seasonal = list(order = c(1, 1, 0), period = 48))

Coefficients:
```

```
        ar1     sar1
     0.9582  -0.4833
s.e. 0.0182   0.0568

sigma^2 estimated as 5.287e+13:  log likelihood = -4604.87,  aic = 9213.75
```

RUTA 2:

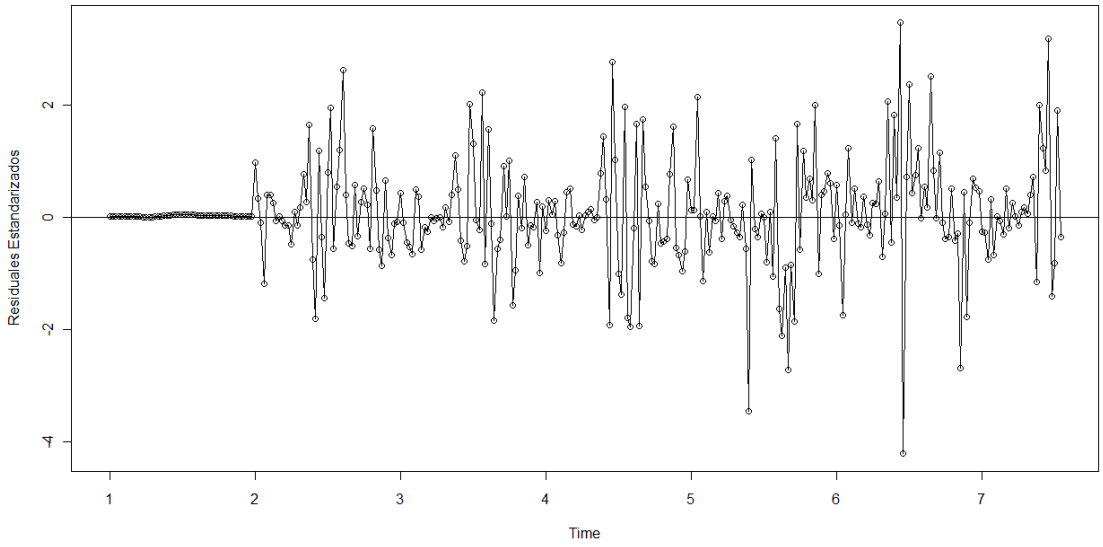
```
m1.jj=arima(jj,order=c(0,1,9),seasonal=list(order=c(0,1,1),period=48))
m1.jj
Call:
arima(x = jj, order = c(0, 1, 9), seasonal = list(order = c(0, 1, 1), period = 48))

Coefficients:
         ma1     ma2     ma3     ma4      ma5     ma6     ma7      ma8
      -0.0541  0.0011  0.1147  0.0338  -0.1195  0.0680  0.0517  -0.0320
s.e.   0.0607  0.0615  0.0624  0.0621   0.0622  0.0638  0.0654   0.0698
         ma9     sma1
      -0.0878  -1.0000
s.e.   0.0631   0.1573

sigma^2 estimated as 3.258e+13:  log likelihood = -4560.84,  aic = 9141.68
```

CHEQUEO DEL DIAGNÓSTICO

A continuación procederemos a comprobar la bondad del afinado de los modelos y para ello, primero hay que observar los residuales estandarizados.

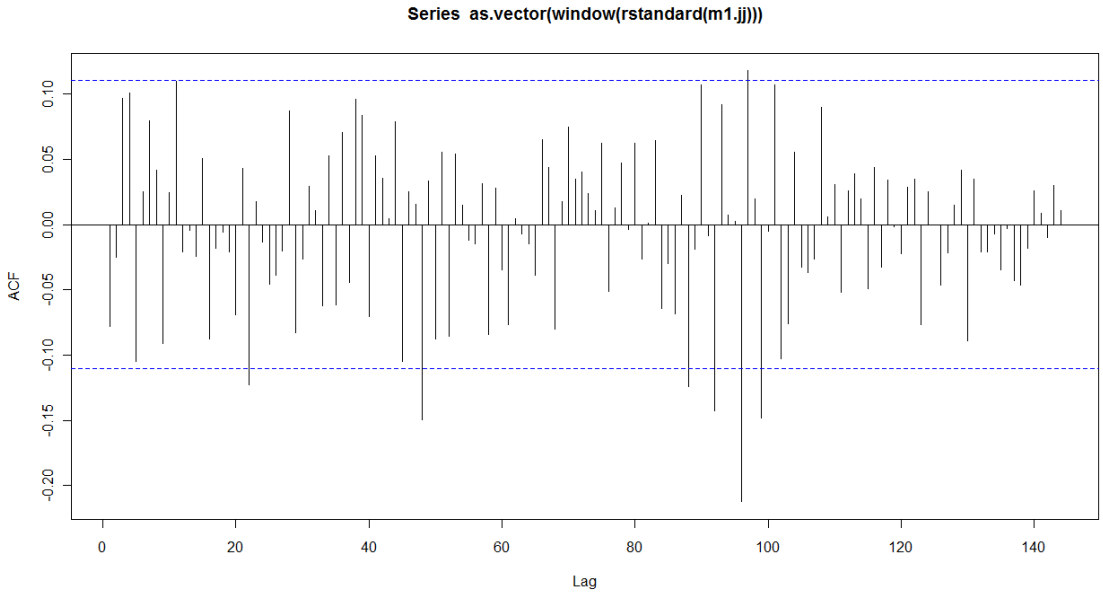
RUTA 1:

```
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)
```



Salvo por una serie de picos a finales de la gráfica, el resto de residuales se encuentran acotados entre 2 y -2 aproximadamente. Para precisar más, visualizaremos el ACF de los residuales.

acf(as.vector(window(rstandard(m1.jj))),lag.max=144)

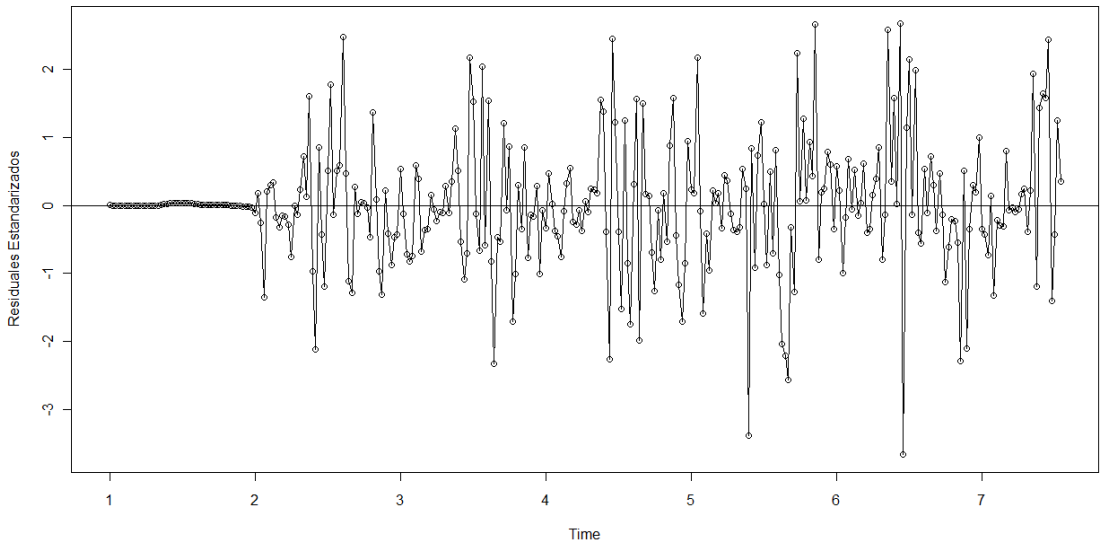


Se aprecian siete correlaciones “estadísticamente significativas”. La correlación más significativa presenta un valor de correlación del entorno del -0,20, un valor no muy elevado.

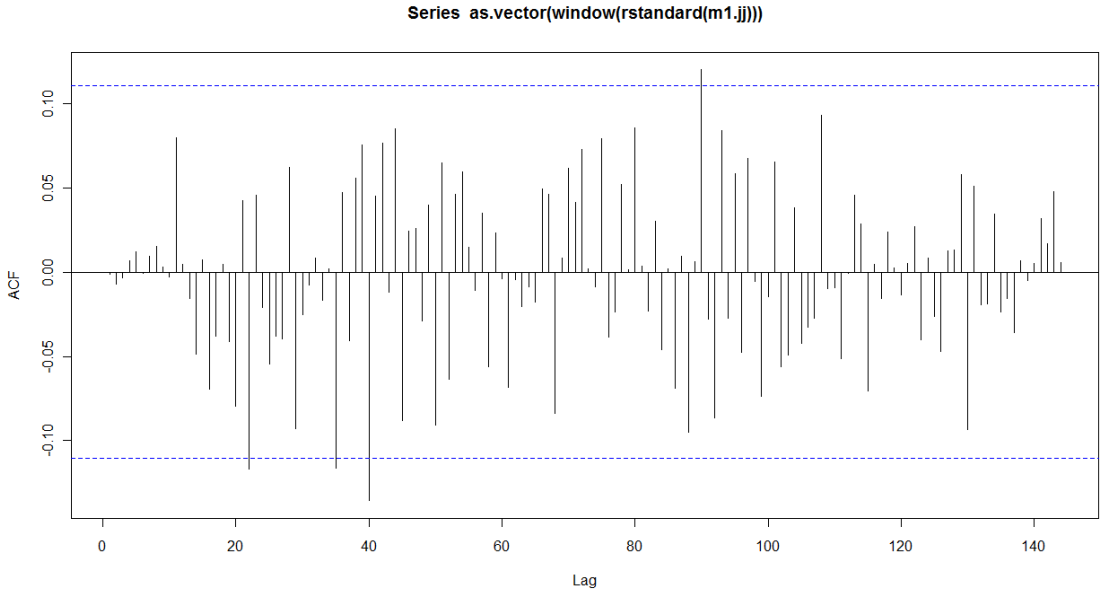
Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

var(jj)
[1] 1.113918e+16
SNR = 1.113918e+16 / 5.287e+13 = 210,69

RUTA 2:
plot(window(rstandard(m1.jj)),ylab='Residuales Estandarizados',type='o')
abline(h=0)



Los picos que presenta la gráfica son aproximadamente los mismos que para la ruta 1, pero se ha reducido la varianza respecto al modelo reducido. Para precisar más, visualizaremos el ACF de los residuales.
acf(as.vector(window(rstandard(m1.jj))),lag.max=144)



El valor de las correlaciones sobresalientes se ha reducido.

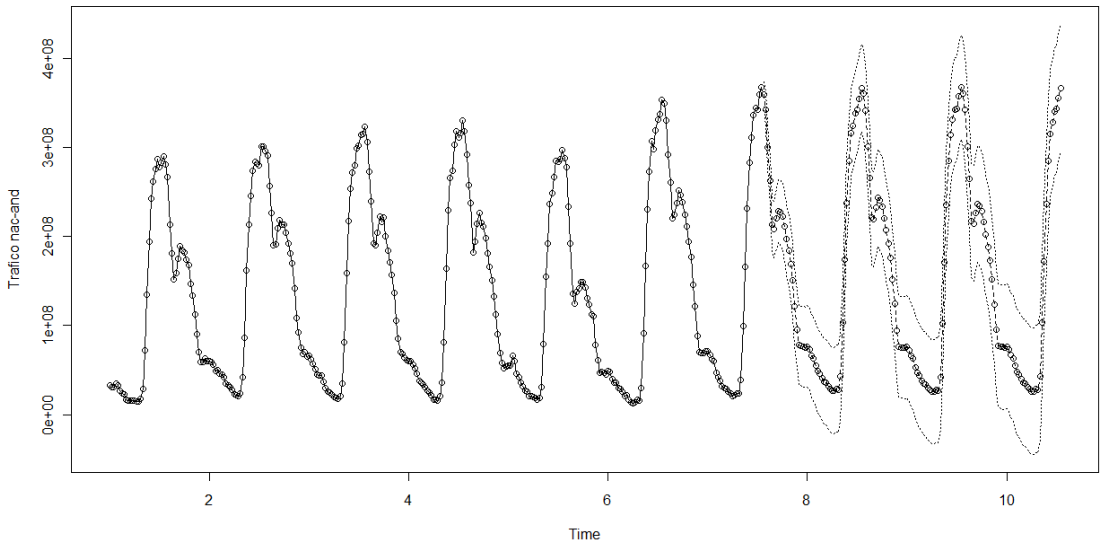
Por otra parte, se puede calcular la relación señal-ruido SNR como la división entre la media de la serie entre la sigma cuadrado (varianza) de las innovaciones del modelo.

var(jj)
[1] 1.113918e+16
SNR = 1.113918e+16 / 3.258e+13 = 341,90

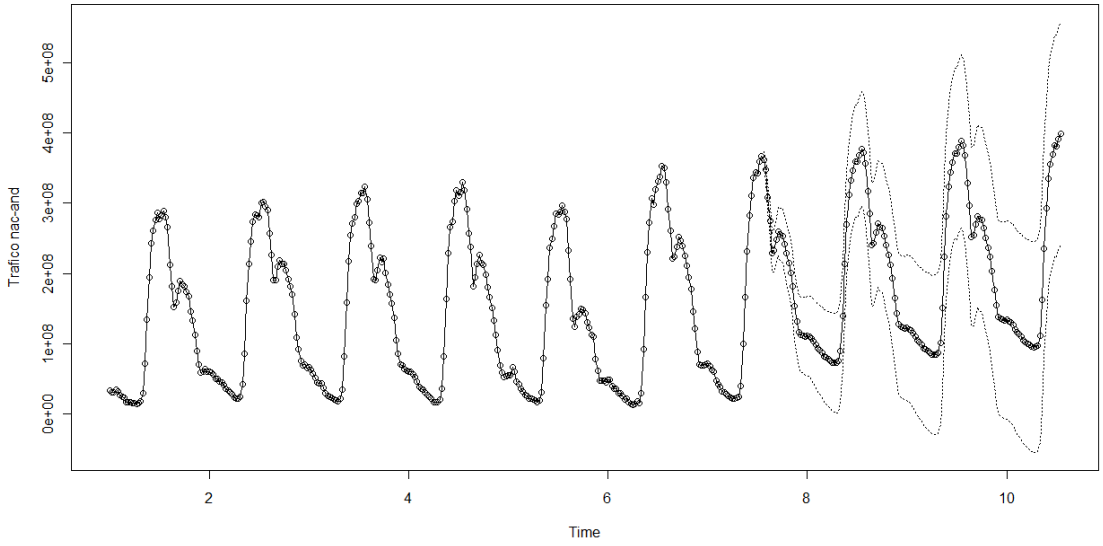
Si comparamos los resultados de ambas rutas, la ruta 2 presenta una mejor relación señal-ruido y unas correlaciones de los residuales más bajas, lo que parece indicar que el modelo tiene una mejor adecuación a la serie temporal. Esto se corresponde con lo indicado por los atributos AIC retornados para los dos modelos, ya que el menor AIC lo presenta el modelo de la segunda ruta. También se ha de destacar que la varianza de los residuales de la ruta 2 es menor que los de la ruta 1.

PREDICCIÓN
plot(m1.jj,n.ahead=144,xlab='Time',type='o',ylab='Trafico nac-and')

ARIMA $(1,0,0)x(1,1,0)_{48}$



ARIMA $(0,1,9)x(0,1,1)_{48}$



A.IV. PCA de los enlaces óptimos de RedIRIS

Esta sección contiene otra prueba del PCA realizado para las series óptimas de RedIRIS.

Éste documento se basa en el análisis realizado en la página:
http://yatani.jp/HCIstats/PCA

CARGA DE LAS SERIES TEMPORALES:

```
and_can_l      =      ts(read.table("c:/Users/Santi/Documents/TFC/Datos
DAT/Partidos/and-can-l-lab.dat"))
gal_ast        =      ts(read.table("c:/Users/Santi/Documents/TFC/Datos
DAT/Partidos/gal-ast-lab.dat"))
mad_nac1       =      ts(read.table("c:/Users/Santi/Documents/TFC/Datos
DAT/Partidos/mad-nac1-lab.dat"))
nac_and        =      ts(read.table("c:/Users/Santi/Documents/TFC/Datos
DAT/Partidos/nac-and-lab.dat"))
nac_clm        =      ts(read.table("c:/Users/Santi/Documents/TFC/Datos
DAT/Partidos/nac-clm-lab.dat"))
nac_ext        =      ts(read.table("c:/Users/Santi/Documents/TFC/Datos
DAT/Partidos/nac-ext-lab.dat"))
nac_gal        =      ts(read.table("c:/Users/Santi/Documents/TFC/Datos
DAT/Partidos/nac-gal-lab.dat"))
```

No se han incluido ni el enlace nac-can-t (porque su modelo es diferente al resto) ni el enlace nac-cat (porque su serie temporal es más corta)

COMBINACIÓN DE LAS ST:
Para crear la matriz de datos que se utilizará para realizar el análisis de componentes principales, se combinan todas las series temporales salvo nac_ext (que se utilizará posteriormente).

```
fin_opt<-cbind(and_can_l,gal_ast,mad_nac1,nac_and,nac_clm, nac_gal)
```

ANÁLISIS DE COMPONENTES PRINCIPALES (PCA) mediante la función prcomp():

```
pca <- prcomp(fin_opt, scale = TRUE)
```

Con resultado:

```
pca
```

```
Standard deviations:
[1] 2.3675082 0.4290433 0.3143731 0.2179033 0.1880343 0.1707564

Rotation:
                PC1          PC2          PC3          PC4         PC5
and_can_1 0.4060555  0.184512075  0.80667957  0.066597125 -0.3551573
gal_ast   0.4107067  0.226523315 -0.42978988  0.713248744 -0.1565866
mad_nac1  0.3895117 -0.895588255  0.05864424  0.105446762  0.1620649
nac_and   0.4128369  0.315318100  0.11914477 -0.007801179  0.8150295
nac_clm   0.4137334  0.114330960 -0.30268304 -0.658413760 -0.0667439
nac_gal   0.4160648  0.008190667 -0.23515332 -0.205310729 -0.3928752
                PC6
and_can_1 -0.14060713
gal_ast   -0.24908778
mad_nac1  -0.07334226
nac_and    0.22717980
nac_clm   -0.53495110
nac_gal    0.75830237
```

Y más concretamente, el porcentaje de varianza explicado por cada componente principal:

```
summary(pca)
```

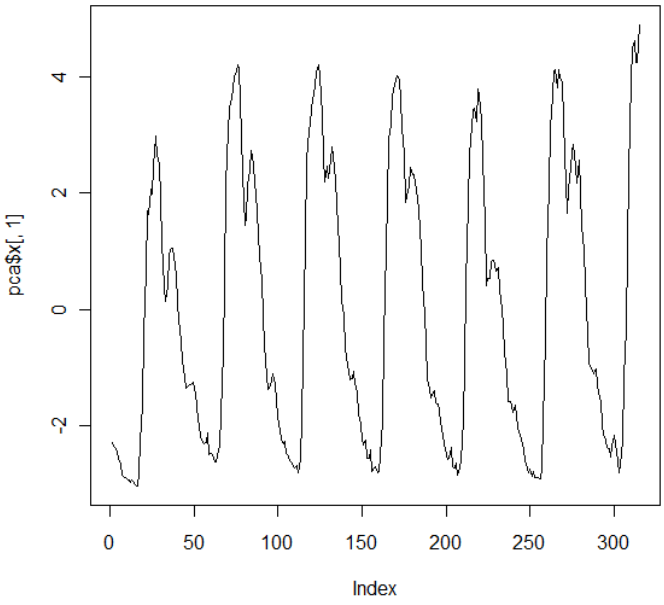
```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6
Standard deviation     2.368 0.4290 0.3144 0.21790 0.18803 0.17076
Proportion of Variance 0.934 0.0307 0.0165 0.00791 0.00589 0.00486
Cumulative Proportion  0.934 0.9649 0.9813 0.98925 0.99514 1.00000
```

Se observa que la primera componente principal ya explica el 93% de la varianza total, lo que significa una alta compresión (para algunos cálculos un 80% ya se considera un buen porcentaje explicativo) y también la posibilidad de reducir el sistema de 5 a 1 sola dimensión. También se cumple el criterio que recomienda la utilización de los componentes principales cuya desviación estándar tenga un valor próximo o superior a 1. En esencia equivale a decir que, a menos que un factor extraiga al menos tanto como el equivalente de una variable original, podemos desecharlo. Este criterio fue propuesto por Kaiser (1960) y es probablemente uno de los más ampliamente utilizados.

Tomando este dato, procedemos a continuación a construir un modelo que permita intentar predecir el comportamiento que tendrá la serie nac_ext que no hemos utilizado para este cálculo. La serie nac_ext es diferente a las series temporales utilizadas para calcular el ACP, aunque sólo mínimamente, con lo que se espera que el ajuste del modelo propuesto y el real de la función sea muy bueno.

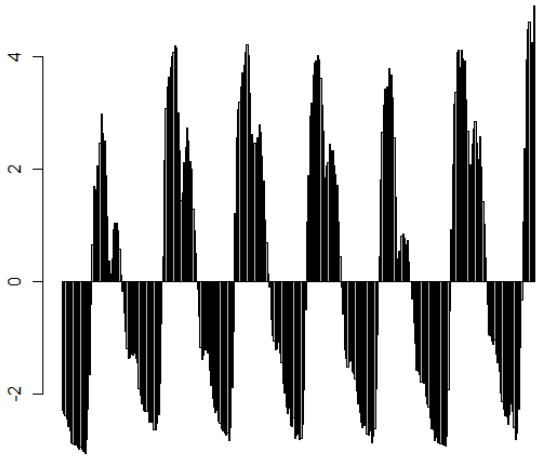
Es una buena idea obtener gráficamente los resultados del análisis:

```
plot(pca$x[,1],type='l')
```



O también:

```
barplot(pca$x[,1])
```



En ambos gráficos se puede ver que la forma del resultado del ACP es bastante predecible.

Una vez realizado este análisis, se utilizarán sus resultados para intentar predecir el comportamiento de la serie nac_ext que se había apartado previamente.

Como nac_ext es una serie que no presenta a priori un patrón claro reconocible (binomial, poisson, etc.), no se especificará la opción “family” de la función y se dejará a la opción por defecto, que es la gaussiana.

```
model <- glm(nac_ext ~ pca$x[,1])
```

```
summary (model)
```

Se obtiene el modelo de la función lineal:

```
Call:
glm(formula = nac_ext ~ pca$x[, 1])

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-8580584  -2056819    146759   1926114  10289550

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 11508349     186059   61.85   <2e-16 ***
pca$x[, 1]   3688889      78714   46.87   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1.090465e+13)

    Null deviance: 2.7363e+16  on 314  degrees of freedom
Residual deviance: 3.4132e+15  on 313  degrees of freedom
AIC: 10354

Number of Fisher Scoring iterations: 2
```

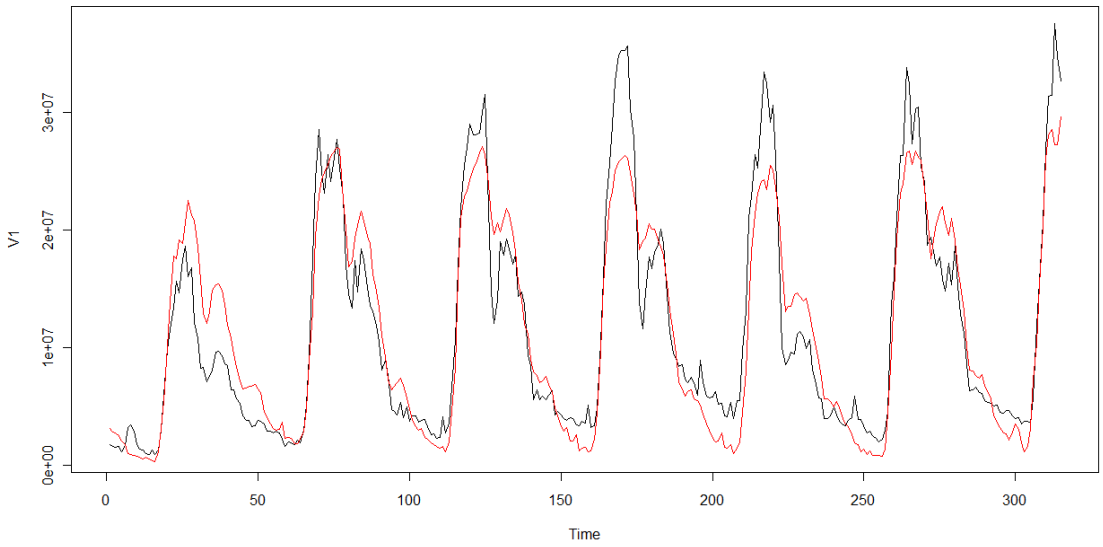
A continuación, se pueden ver los valores esperados para la serie nac_ext:

```
fitted(model)
```

```
        1         2         3         4         5         6
3103621.3 2870470.0 2743776.0 2563815.5 2053221.2 1745909.0
        7         8         9        10        11        12
 984612.9  924291.3  863014.4  799811.4  681644.9  514115.3
…
```

Para tener una idea más clara de la afinidad de la estimación, es mejor trazar un gráfico que superponga la serie nac_ext con su modelo predictivo (serie original en negro y serie predictiva en rojo):

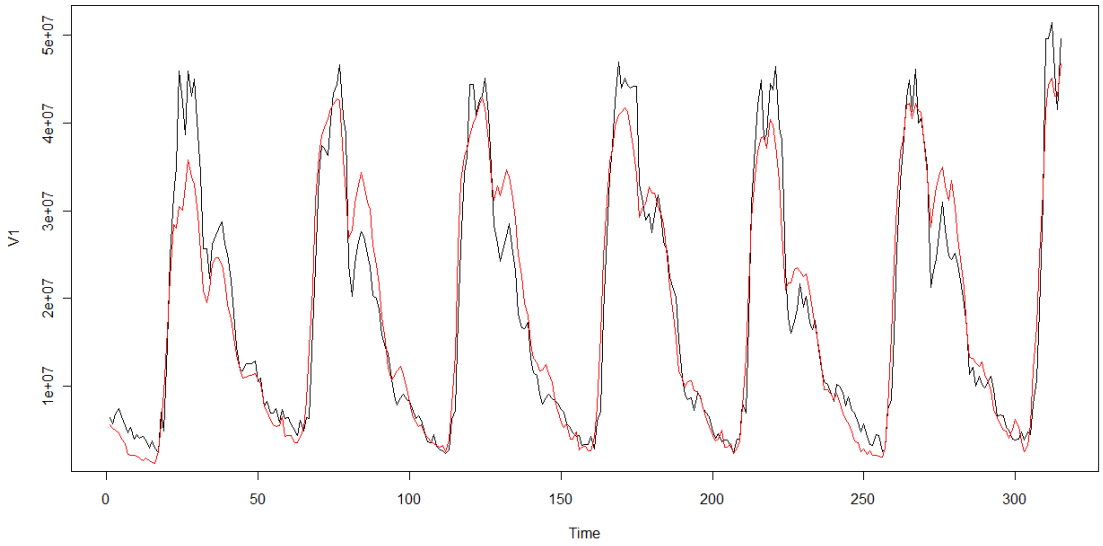
```
plot(nac_ext)

lines(fitted(model), col='red')
```



Teniendo en cuenta que no se ha utilizado a la serie nac_ext para el ACP, el resultado no es demasiado malo y ajusta bastante bien a la función, al menos en su comportamiento general.

Si se repite la operación para la serie and_can_l que sí que se ha utilizado para el cálculo del ACP, el resultado es el siguiente:

```
plot(and_can_l)

model2 <- glm(and_can_l ~ pca$x[,1])

lines(fitted(model2), col='red')
```



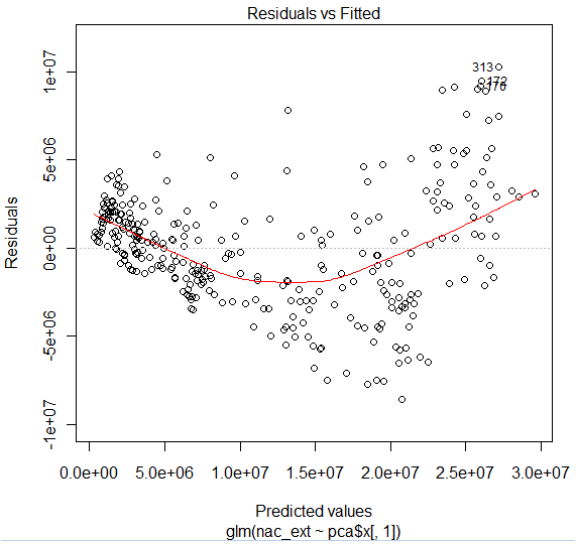
También se puede observar los datos concretos de afinidad proporcionados por la función summary:

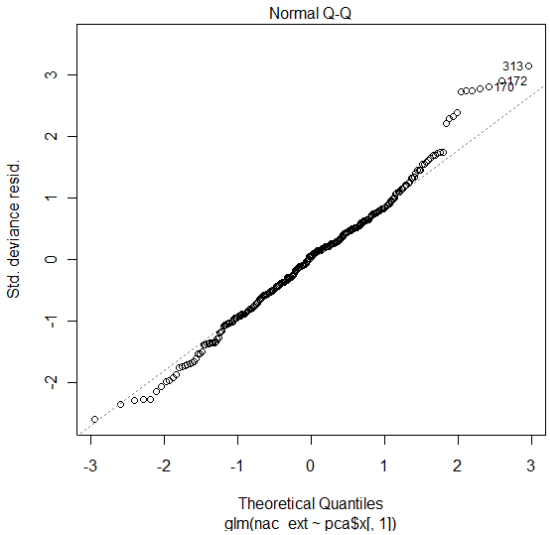
```
summary(model2)
```

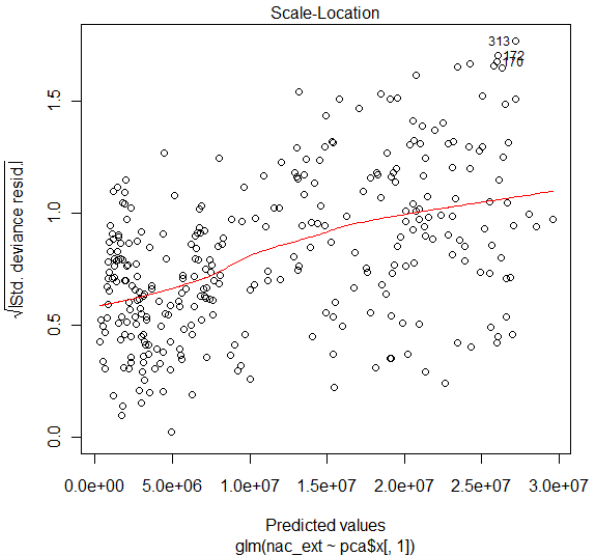
```
Call:
glm(formula = and_can_1 ~ pca$x[, 1])

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-9028071  -2067055    329769   2011369  15417942

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 18634301     220125   84.65   <2e-16 ***
pca$x[, 1]   5751865      93125   61.77   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1.526333e+13)

    Null deviance: 6.3005e+16  on 314  degrees of freedom
Residual deviance: 4.7774e+15  on 313  degrees of freedom
AIC: 10460

Number of Fisher Scoring iterations: 2
```

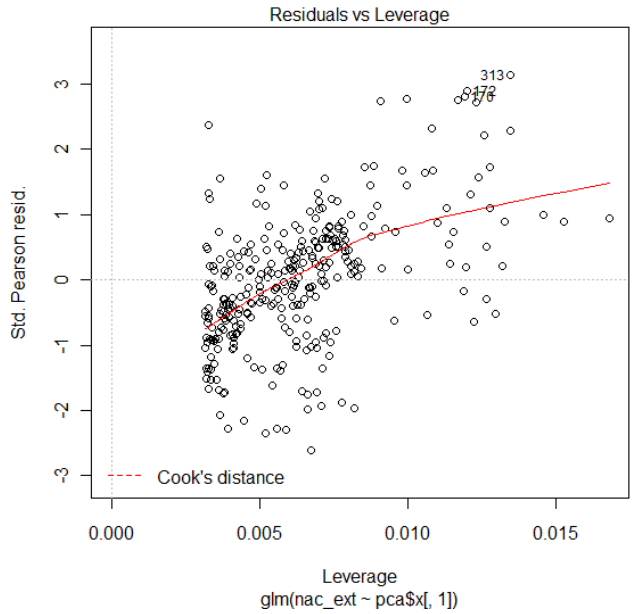
Volviendo al caso de nac_ext, aun se pueden extraer aun más datos del modelo:

```
plot (model)
```




Normal Q-Q
Std. deviance resid.
Theoretical Quantiles
glm(nac_ext ~ pca$x[, 1])


Scale-Location
√|Std. deviance resid.|
Predicted values
glm(nac_ext ~ pca$x[, 1])


Residuals vs Leverage
Std. Pearson resid.
Cook's distance
Leverage
glm(nac_ext ~ pca$x[, 1])

Los 4 esquemas surgen mediante el mismo comando y se suceden clicando sobre el gráfico. En orden de aparición, tenemos:

- Diagrama Residuales vs. Valores Predictivos: muestra la media de residuales asociada a cada valor que aparece en la serie temporal. Los residuales son la diferencia entre el valor real acontecido y el valor predicho. Como se puede ver, la curva dibujada sugiere una distribución normal típica de los eventos aleatorios.

- Diagrama QQ entre la desviación estándar de los residuales y las cantidades teóricas: muestra el ajuste entre las desviaciones estándar registradas y las desviaciones esperadas para una distribución normal teórica. La recta teórica representa la trayectoria que presentaría un ajuste exacto entre la desviación estándar registrada y las cantidades teóricas (representativa de una distribución normal exacta).

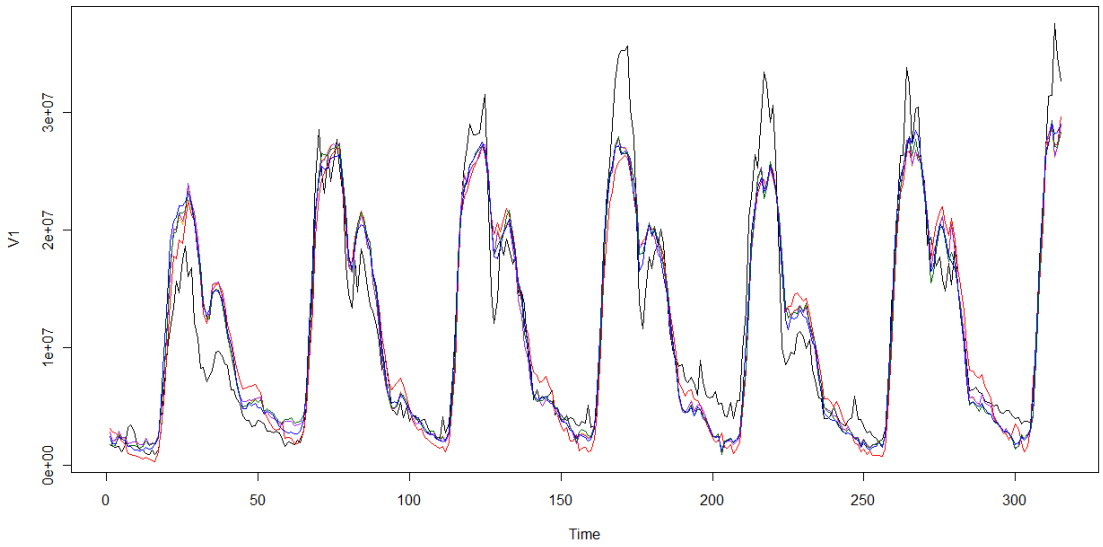
- Diagrama escala-localización: muestra la comparativa entre la raíz de las desviaciones estándar contra la escala de valores de la serie predicha. Se puede ver que cuanto mayor es el valor predicho, mayor es su varianza asociada.

Una vez concluido el análisis, se puede repetir para distintas cantidades de componentes principales. A medida que se incrementa la cantidad de componentes principales incluidos, aumentará la fidelidad de la predicción del modelo a costa de un mayor coste computacional y de una simplificación menor del modelo. Así, se pueden calcular distintos modelos en función de la cantidad de componentes principales incluidos para la regresión:

```
model1 <- glm(nac_ext ~ pca$x[,1])
model2 <- glm(nac_ext ~ pca$x[,1]+pca$x[,2])
model3 <- glm(nac_ext ~ pca$x[,1]+pca$x[,2]+pca$x[,3])
model4 <- glm(nac_ext ~ pca$x[,1]+pca$x[,2]+pca$x[,3]+pca$x[,4])
model5<-glm(nac_ext~pca$x[,1]+pca$x[,2]+pca$x[,3]+pca$x[,4]+pca$x[,5])
```

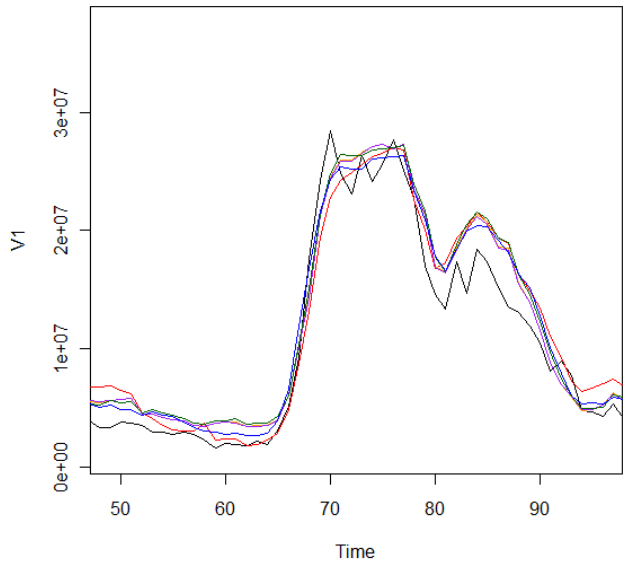
Si a continuación se representan las cinco predicciones juntas en un mismo gráfico junto al modelo real, resulta mucho más sencillo ver la importancia de la aportación de cada componente a la predicción final:

```
plot(nac_ext)
lines(fitted(model1), col='red')
lines(fitted(model2), col='orange')
lines(fitted(model3), col='purple')
lines(fitted(model4), col='darkgreen')
lines(fitted(model5), col='blue')
```



Debido a la gran varianza capturada por el primer componente principal, las aportaciones del resto de los componentes principales se ven relegadas a un rizado añadido a la estructura establecida por la primera componente. Por otra parte y como indica la teoría, estas pequeñas aportaciones afinan mejor la forma real de la gráfica. Esto se puede ver con más detalle si se restringe la zona observada a una pequeña sección de la serie estudiada como por ejemplo a un solo día (recordemos que la frecuencia de la serie es de 48 muestras/día):

```
plot(nac_ext,xlim=c(49,96))
lines(fitted(model1), col='red')
lines(fitted(model2), col='orange')
lines(fitted(model3), col='purple')
lines(fitted(model4), col='darkgreen')
lines(fitted(model5), col='blue')
```



En esta gráfica se observan mejor las aportaciones de cada componente principal a la predicción del modelo inmediatamente anterior.